Received: 2 May 2022 | Revised: 14 February 2023 | Accepted: 2 April 2023

DOI: 10.1002/smr.2570

RESEARCH ARTICLE - TECHNOLOGY

Journal of Software: Evolution and Process

WILEY

Software medical device maintenance: DevOps based approach for problem and modification management

Maria Raffaella Martina[1] | Elisabetta Bianchini[1] | Sara Sinceri[1] | Martina Francesconi[2] | Vincenzo Gemignani[1]

[1]Institute of Clinical Physiology (IFC), National Research Council (CNR), Pisa, Italy

[2]Department of Clinical and Experimental Medicine, University of Pisa, Pisa, Italy

Correspondence
Maria Raffaella Martina and Vincenzo Gemignani, Institute of Clinical Physiology (IFC), National Research Council (CNR), Pisa, Italy.
Email: martinamariella@gmail.com and vincenzo.gemignani@ifc.cnr.it

Abstract

Software problem resolution and change control processes are crucial for the performance, safety, and regulatory clearance of a medical device. Software changes are frequent throughout the whole life cycle of a product; therefore, iterative analysis and risk assessment are necessary to determine the impact of changes on the whole system. IEC standard 62304 clarifies software medical device life cycle processes including problem and modification analysis and requirements. The aim of this work is to describe a cost-effective and fully customizable solution for software medical devices problem and modification management implemented in our group. Key features of the problem and modification management related to the software life cycle have been identified and taken into account. A digital commercial platform for software development and maintenance has been identified and used to implement an architecture satisfying the standard. The architecture solution (SWMA, software maintenance architecture) has been successfully developed and a description of how it addresses IEC 62304 standard requirements both in terms of problem resolution and in terms of change control has been provided. The presented system can be useful for manufacturers/groups to establish software maintenance plans that include activities and tasks related to software problem resolution and change control processes. The approach can be adopted for analyzing and resolving problems arising before and after the release of a medical device.

KEYWORDS
change control, DevOps approach, IEC 62304 standard, medical device, software life cycle, software problem resolution

1 | INTRODUCTION

The digital revolution has been impacting each field of medicine. Most of medical devices (MDs) are dependent on software for their processes and controls as well as for their design and development. Recently, the development of medical device software (MDSW), as defined by the

Maria Raffaella Martina and Elisabetta Bianchini contributed equally to the paper.

Guidance on Qualification and Classification of Software in Regulation (EU) 2017/745 – MDR,[1] is increasingly helping physicians to make accurate diagnosis and guide decisions, highlighting the high potential for innovation and improvement in the health care sector.

MDSWs play a relevant clinical role impacting on patients and operators, thus need to be highly regulated. In most countries, regulation establishes criteria and requirements to be met for medical device software to be marketed.[2] In particular, US Food and Drug Administration (FDA) regulates the medical device sector in the United States,[3] whereas the Medical Device Regulation (MDR 2017/745), in force from May 2021 postponed by 1 year due to the coronavirus pandemic,[4] regulates the medical device sector in the EU countries replacing the previous Directive 93/42/EEC[5] and EU MDD (Medical Device Directive) 2007/47/EC.[6]

Medical device manufacturers marketed in the EU and European Economic Area must comply with the new MDR that focuses on continuous clinical evaluation processes well-structured in procedures and documented with detailed report. Furthermore, processes and documentation relating to post-market clinical follow-up evaluation, summary of safety and clinical performance, risk management, and post-market surveillance must be organized according to the risk class of the device. The MDR establishes that the manufacturer must have specific procedures to confirm compliance with the general safety and performance requirements.[7] One of the changes introduced by the new regulation concerns the software as a medical device sector. In particular, rule 11 makes the certification process clearer by classifying most of the medical software starting from class IIa up to III.[8] A "Medical Device Software" is a software intended by the manufacturer to be used, alone or in combination, for human beings, including diagnosis, prevention, monitoring, prediction, prognosis, treatment, or alleviation of disease.[1] In particular, a software is subject to MDR if it performs operations on data other than storage or archiving and if these operations are for medical purposes and for the benefit of the patient's health.[2]

International organizations (such as ISO - International Organization for Standardization), deliver a set of standards to guide medical device manufacturers to meet complex regulatory requirements. An entity can use, to demonstrate to the regulatory authorities the safety and reliability of its processes and products, the compliance with the standards related to its activity. A key standard describing processes and requirements for a safe design and maintenance of the medical device software is the IEC 62304 Medical device software - Software life cycle processes[9] published by the International Electrotechnical Commission (IEC) and recognized by the FDA in USA as well. The requirements of IEC 62304 directly relate to some of the requirements of other standards, for example, ISO 13485 concerning the quality management standard for medical devices, the risk management standard ISO 14971, and the medical device usability standard IEC 62366-1.[10] Furthermore, within the quality management system an efficient documentation control must be provided for all the processes of the medical device life cycle, ranging from development to maintenance and post-market activities,[11] as specified by the new MDR that emphasizes the value of the interconnection between the documented management system, risk analysis, safety, and performance of medical devices.

IEC 62304 clarifies software life cycle processes for management, implementation, integration as well as problem and modification analysis. Software problem resolution and change control processes are crucial for the performance, safety, and regulatory clearance of a medical device. Changes are continuously implemented throughout the life cycle of a software, including new features or improvements that might impact on the software use. Thus, iterative analysis and risk assessment are necessary to determine the effects of changes on the whole system.[12]

IEC 62304 explores the issue of problem resolution by specifying that for each problem encountered in the software life cycle, processes must be activated to make decisions, take correct actions, assign tasks, and compile related documentation. To manage this complex network of regulatory requirements and processes, new approaches can be useful, such as DevOps where development and operations are integrated activities to rapidly manage tasks, ensure quality and safety, control feedback, and reduce development time and costs thanks to digital tools and workflows.[13]

In this paper we propose a cost-effective, replicable and fully customizable solution for software medical devices problem and modification management, based on a low-cost commercial software platform, that was implemented in our group in a spin-off project of the CNR Institute of Clinical Physiology and University of Pisa (Quipu srl). In the subsequent sections, the adopted digital commercial platform, the implementation of the architecture solution (SWMA) and how this can satisfy IEC 62304 standard requirements are presented.

1.1 | Empirical evaluation

The DevOps is considered a cultural approach able to provide greater collaboration and communication by aligning people, processes, and tools with benefits, for example, for multidisciplinary teams involved in the entire life cycle of a product. Some DevOps tools[*] are commercially available, and companies can adopt them in order to increase time efficiency and to reduce time to market. DORA 2019 State of DevOps report[14] shows that teams using DevOps tools and practices to build, test, and deliver software are able to release deliverables more frequently (208 times more frequently and 106 times faster than low-performing teams), with higher quality and stability.

Regarding the literature related to the DevOps methodology, a need for more empirical studies in this area is reported.[15] Hence, recent papers investigated, by means of interviews or questionnaires, the use of the DevOps approach in software organizations[16] and report as

[*]https://www.atlassian.com/devops.

output that this methodology is increasingly in use for different goals related, for example, to reduce lead-time, improve problem-solving, and feedback collection.[17] In all cases, DevOps promotes communication and team collaboration thus helping to create bridges to overcome the barriers between development and operations skills as well as to improve quality, safety, and effectiveness throughout the software lifecycle.[18]

The perception reported in some cases was the lack of data relating to benefits, costs, and risks not sufficiently explored, and some organizations have introduced new employee role called DevOps Engineer to implement and monitor the DevOps methodology.[17] Even more so it is necessary to spread the use of low-cost tools and customizable implementations to improve task coordination, interoperability of the team, and thus quality, traceability, and safety of the products to meet the challenges associated with iterative and continuous integration, development, deployment, test, track, and release phases for any software.

In our group, we experienced advantages thanks to the implementation of a DevOps approach mainly related to the maintenance of the developed software and of its medical device file. A software is not a static system, it changes dynamically, and more than one software version is released every year due to bug fixes and improvements, to meet new user needs and new cybersecurity requirements. One of the crucial issues for the maintenance of software life cycle processes is tracking every action while keeping documentation up to date. This would be very complex without the implemented architecture that allows tracking of all software changes and how they affect different aspects of the medical device file, for example, in changing a requirement, a test to perform, the risk analysis, and user documentation. It would be difficult to manage the links among all different aspects covered by the medical device file without the help of the developed architecture. In our team, this technology is beneficial mainly for interoperability and traceability, that is, having tools to effectively track the changes on the affected releases is essential as well as easily create and export the documentation updated for each version.

2 | METHODS

Key aspects of problems/modifications' management related to the software life cycle were identified and taken into account. These include identification of a digital platform and requirements' identification based on available standards. A digital commercial platform for software development and maintenance was identified to develop our customized architecture solution addressing medical device's requirements.

2.1 | Digital platform used for the development of an architecture solution

Digital platforms for managing software development and maintenance, besides safety and security, need to take into consideration other users' needs, ranging from planning of tasks to tracking of data processes to provide a heuristic management within the software life cycle; thus, fundamental requirements to be met are

- high customization
- possibility of integrating with other platforms
- cost-effective

Based on the three points above, our customized digital solution was implemented by using Jira (https://www.atlassian.com/software/jira), a commercial software platform developed by Atlassian Corporation Plc (Sydney, Australia). Jira is a software development tool built to help teams to manage their work in terms of planning, tasking, tracking, reporting, and team interaction bringing the advantage of managing all activities in an integrated and smarter way, thus improving the organization of work.

The DevOps approach is the combination of software development practices and operations with the goal of optimizing product quality and delivery to users. However, many companies find difficult to evaluate their performance in order to enhance them. Jira software allows to measure and improve organization's processes thanks to features that allow manufacturers to visualize and measure progress from idea through to production, based on information and data on product distribution easily available providing elevated control and visibility in the company's toolchain.[19]

2020 DevOps Trends Survey among 500 developers and IT decision makers reports that most of the respondents said DevOps had a direct impact on business metrics but encountered barriers in their DevOps implementation. In this context, Jira software can be considered a useful tool for manufacturers to support the implementation of DevOps approach in their company.[20]

This platform presents high degree of customization and configurability that can be exploited by internal expertise and know-how. Furthermore, Jira has an affordable cost and can be easily integrate with other platforms for a transversal tasks' management.

2.1.1 | Documentation control and problem and modification management integration

Documentation control is a crucial point for the whole life cycle of a medical device including development, maintenance, post-market surveillance, and control of records in order to ensure quality and safety. Furthermore, documentation management is a valuable control tool to ensure continuous improvement and risk minimization as fundamental aspects of the medical device design and deployment.

Jira can connect with Confluence,[†] a commercial software platform developed by Atlassian Corporation Plc (Sydney, Australia) used by our group for documentation control. A scheme for Jira and Confluence integration system is reported in Figure 1. Our group can rely on a structured quality management system, which is relevant to design and develop medical software. Quality management system manages documentation and records related to the key company's activities through a customized architecture based on Confluence, supporting features such as implementation, versioning, approval, secure access of documentation, and compliance with standards throughout the software life cycle.

Confluence–Jira integration in our system simplifies the management of the software development and maintenance processes because the two applications work together from planning to execution of the activities with organized workflows and task tracking by integration of the documentation management. Our team members are in sync in all activities leading up to releases and updates, with a centralized space to organize requirements, release notes, goals, and tasks to be done. These software platforms translate product requirements and user stories into specific actionable features, thus simplifying context switching with direct links between documentation and related issues in Jira. Furthermore, Jira and Confluence softwares help ease communication between development teams and non-technical stakeholders and bring the advantage of having a single source of truth for software documentation. (https://www.atlassian.com/software/confluence/jira-integration).

2.2 | Requirements' identification

To fulfill the regulatory requirements for the analyzed processes, we referred to IEC 62304, the reference standard for software medical devices. Compliance with this standard requires manufacturers to establish software maintenance processes that include problem and modification analysis relating to feedback, verification of reported problems, and management and approval for implementing a modification option. Software problem resolution and change control processes are crucial for the performance, safety, or regulatory clearance of a medical device and are mentioned in different points of the IEC 62304 standard (i.e., Clauses 5, 6, 8, and 9).

Moreover, as defined in Clauses 4.2 and 4.3, a software system is assigned by the manufacturer to safety class A, B, or C according to possible events on patients, operators, or other subjects involved resulting from a hazard connected to the software system. In particular, class A when no injury or damage to health is possible, class B if non-serious injury is possible, and class C in case of death or serious injury is possible. Modifications control is crucial to manage the risk management process complying with regulation and international standards (i.e., ISO 14971).

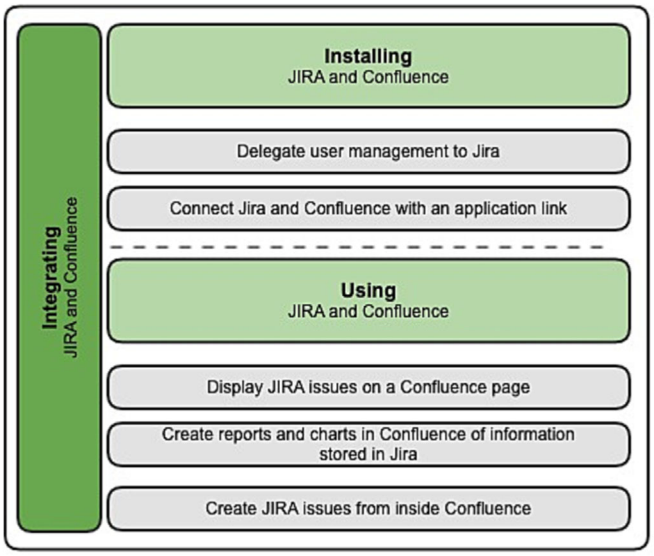

Integrating
JIRA and Confluence
Installing
JIRA and Confluence
Delegate user management to Jira
Connect Jira and Confluence with an application link
Using
JIRA and Confluence
Display JIRA issues on a Confluence page
Create reports and charts in Confluence of information stored in Jira
Create JIRA issues from inside Confluence

FIGURE 1 Jira and Confluence integration system and processes.

[†]https://www.atlassian.com/software/confluence.

A standard revision project proposes to extend the scope of IEC 62304 beyond medical devices to health software used to manage, preserve, or improve people's health including cybersecurity issues in medical devices and health software.[10] Regulatory cybersecurity guidance draft for manufacturers of medical devices has been published in the United States.[21] In EU, cybersecurity requirements are reported in Annex I of the MDR,[22] deal with both premarket and post-market aspects, with the aim to provide a basis for the development of recommendations and guidance for medical device manufacturers as well as to manage cybersecurity risks and protect patient health information. This topic needs further study and will not be covered in this paper.

The next sub-sections report a detailed description of the requirements' identification for the two key processes addressed within this work: software problem resolution and change control.

2.2.1 | Software problem resolution process requirements

The main phases to be managed in the software problem resolution process are specified in Clause 9 of IEC 62304 standard. Due to its relevant role in the development and maintenance cycle, this process is mentioned also in other relevant steps of the standard (Figure 2). The software problem resolution process must be defined for handling problems detected in the software at each stage of the life cycle.

The software configuration management planning specifies when the manufacturer should use the problem resolution process to manage anomalies, integration, and integration testing.

Problems can be discovered before or after release, inside the manufacturer's organization, or outside it. Manufacturers establish software maintenance plans with activities and tasks for the use of the problem resolution process for analyzing and solving problems arising after release of the product.

All the problems detected during the life cycle of the software medical device must follow a path of investigation and change control that will be tested and recorded in problem reports including their resolution and verification. Problem reports are evaluated to determine how they affect the safety of a released software product and whether a change to the released software product is needed to address the problem.

With reference to point 9 of the IEC 62304 standard, the problem reports are classified according to the type (e.g., corrective, preventive, or adaptive to new environment), scope (e.g., size of change, number of device models affected, and supported accessories affected), and criticality regarding effect on performance, safety, security, or regulatory clearance of a medical device.

The manufacturer investigates the problem to assess the relevance to safety by referring to the software risk management process and documents the results, then creates a change request for actions needed to correct the problem, or documents the rationale for taking no action. The manufacturer may also decide not to correct the problem if it is not relevant to safety.

The verification of software problem resolution will determine if the problem has been resolved, the problem report has been closed and adverse trends reversed, as well as if additional problems have been introduced to correct a problem or implement a request.

After all testing is completed, the test documentation will include the outcomes, anomalies, software version tested, hardware and software test configurations, relevant test tools, date tested, and identification of the tester.

It is particularly important to verify that the risk control measures integrated into the medical device are not negatively modified by a software change. It is also crucial to assess the impact of a software change in terms of new hazards or modifications of risks' mitigation as this could even change the software safety classification.

IEC 62304 standard discerns between software maintenance (Clause 6) and software problem resolution (Clause 9) processes. The core of the software maintenance process is defined as the reception of feedback after release of the software product to ensure that safety-related problem reports are addressed and reported to appropriate regulatory authorities and affected users. Moreover, the maintenance process provides that the implemented software is revalidated and re-released. On the other hand, the core of the software problem resolution process is analyzing problem reports, deciding on changes and evaluating the consequences, then implementing and verifying the changes while maintaining the consistency of the software configuration items including the risk management file. It follows that the software maintenance process uses the operating software problem resolution process to analyze problems' report to produce changes' requests that identify the configuration items to change and the verification steps.

2.2.2 | Change requests control requirements

Change request is a documented specification of a change to be made to a software product. As specified in Clause 6 of IEC 62304, the manufacturer evaluates each change request for its effect on the organization, released software products, and interfaced systems and decide on approval. Due to its relevant role in the development and maintenance cycle, requirements related to change requests management are mentioned also in other relevant steps of the standard.

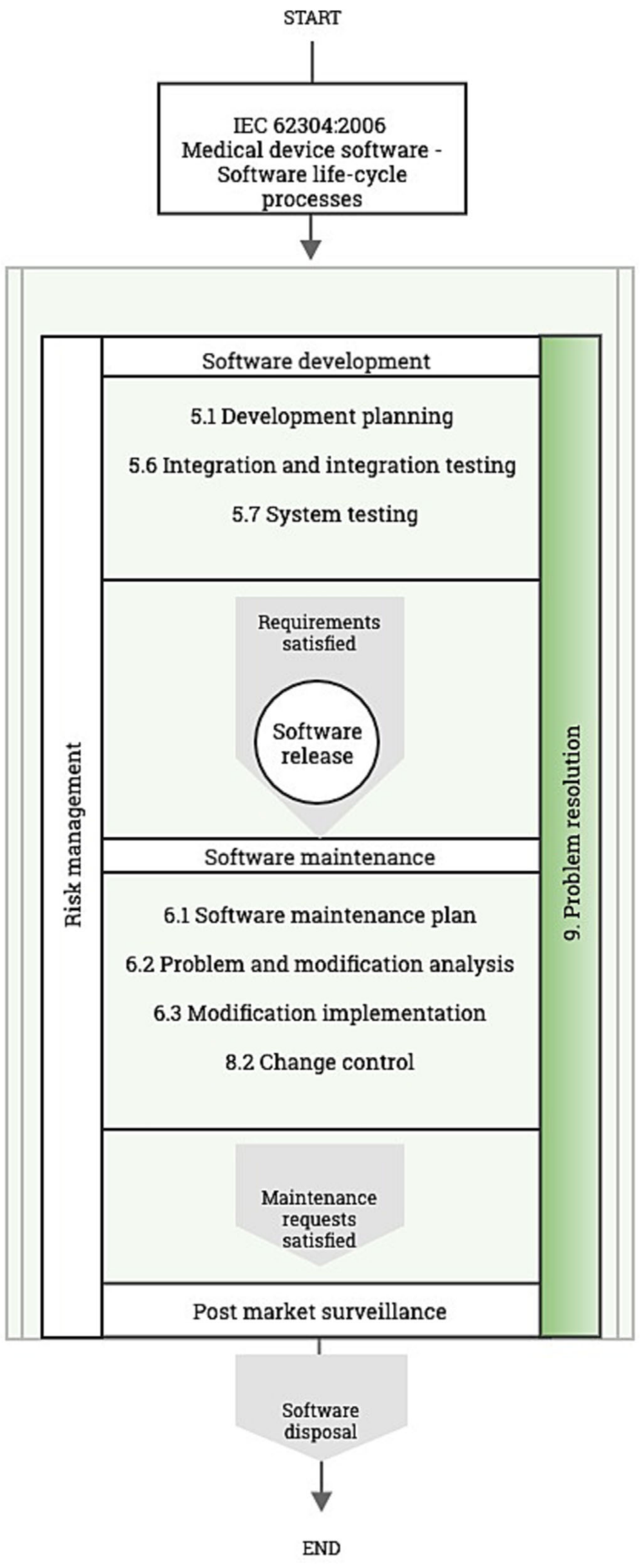

START
IEC 62304:2006 Medical device software - Software life-cycle processes
Software development
5.1 Development planning
5.6 Integration and integration testing
5.7 System testing
Requirements satisfied
Software release
Software maintenance
6.1 Software maintenance plan
6.2 Problem and modification analysis
6.3 Modification implementation
8.2 Change control
Maintenance requests satisfied
Post market surveillance
Risk management
9. Problem resolution
Software disposal
END

FIGURE 2 Software problem resolution application within the life cycle processes of a software medical device according to the international standard IEC 62304:2006.

The manufacturer identifies the approved change requests and, if needed, informs users and regulators about the problems detected in the software and the consequences of using the unmodified version and explains the type of changes and how to receive and install them.

The approved changes are implemented as specified in the change request with change control process entered into established maintenance or software development process and the manufacturer releases the changes as part of a full re-release of the software or as a modification kit. Within the changes' implementation, the manufacturer undertakes all the necessary actions to verify the changes including changes to the software safety classification. The traceability of the changes is crucial, and it is achieved by means of an audit trail where each change request, relevant problem report, and approval of the change request can be traced.

The manufacturer is responsible for controlling and documenting the activities and information relating to the change request. This activity is necessary to avoid unauthorized or unintended changes to the software configuration items and to ensure the full implementation and verification of the approved change requests. A change needs to be linked to an approved and documented change request.

All the described requirements, mentioned in Clauses 5, 6, 8, and 9 of IEC 62304, have been addressed, and the related implementations are shown in the following section.

3 | RESULTS

3.1 | Developed architecture solution

The implemented solution SWMA based on Jira platform has been designed to fulfill the reference requirement both in terms of problem resolution and in terms of change control by addressing the activities described below.

3.1.1 | Inputs to the software problem resolution process

Sources of software problem reports may include

- anomalies detected during software development, internal testing, external testing or by the final user;
- software feedback evaluation; and
- non-conformity reports and/or customer complaints,

as summarized in Figure 3.

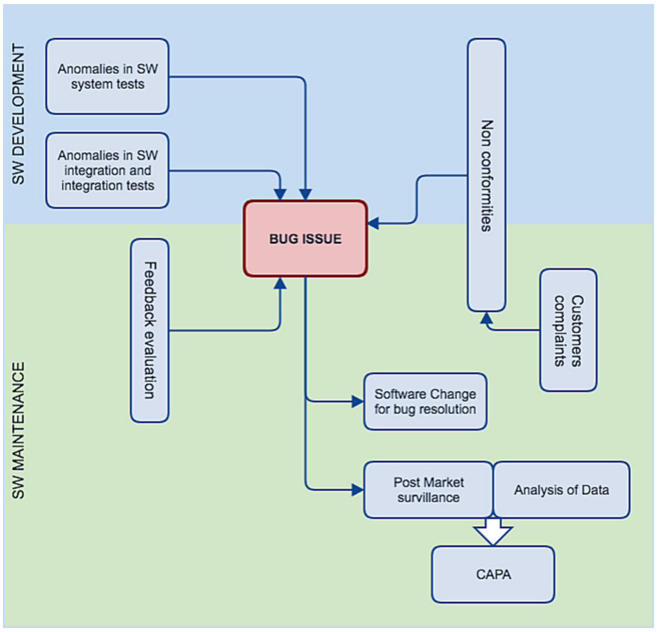
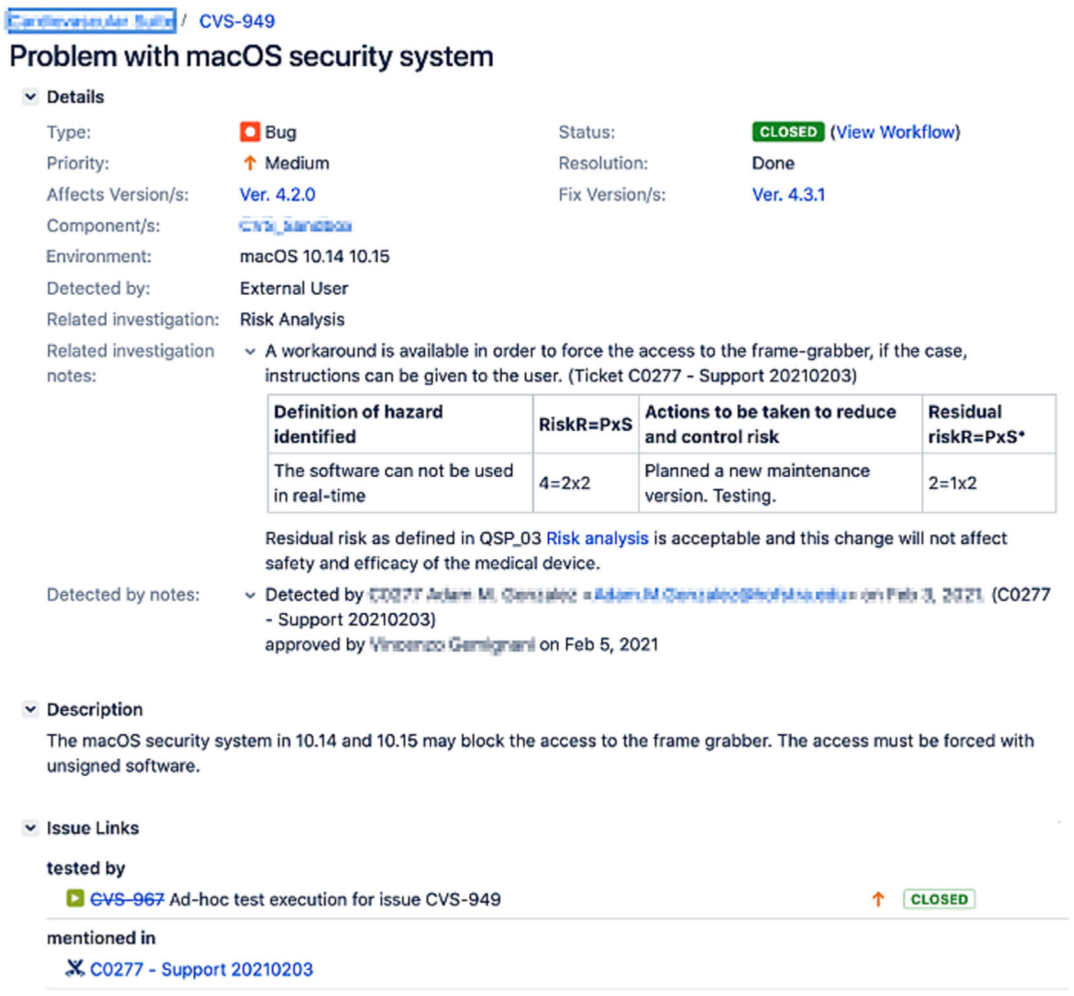


FIGURE 3 Inputs and outputs to software problem report.



[redacted] / CVS-949

Problem with macOS security system

Details

Type:	Bug	Status:	CLOSED (View Workflow)
Priority:	Medium	Resolution:	Done
Affects Version/s:	Ver. 4.2.0	Fix Version/s:	Ver. 4.3.1
Component/s:	[redacted]		
Environment:	macOS 10.14 10.15		
Detected by:	External User		
Related investigation:	Risk Analysis		

Related investigation notes: A workaround is available in order to force the access to the frame-grabber, if the case, instructions can be given to the user. (Ticket C0277 - Support 20210203)

Definition of hazard identified	RiskR=PxS	Actions to be taken to reduce and control risk	Residual riskR=PxS*
The software can not be used in real-time	4=2x2	Planned a new maintenance version. Testing.	2=1x2

Residual risk as defined in QSP_03 Risk analysis is acceptable and this change will not affect safety and efficacy of the medical device.

Detected by notes: Detected by [redacted] (C0277 - Support 20210203)
approved by [redacted] on Feb 5, 2021

Description

The macOS security system in 10.14 and 10.15 may block the access to the frame grabber. The access must be forced with unsigned software.

Issue Links

tested by

~~CVS-967~~ Ad-hoc test execution for issue CVS-949 CLOSED

mentioned in

C0277 - Support 20210203

FIGURE 4 Example of bug issue in Jira managing problem.

TABLE 1 Status and description of a bug issue.

Status	Description
OPEN	The bug has been recorded and must be investigated.
IN PROGRESS	The bug has been investigated and its resolution is in implementation.
RESOLVED	The bug resolution has been implemented and must be approved.
CLOSED	The bug has been closed. It may be "Resolved" or "Unresolved" (Resolution field).
REOPENED	The bug has been re-submitted.

Problems detected from different sources and affecting the product shall be documented as "bug" issues in Jira (Figure 4), according to the internal risk management procedure.

A responsible is identified for collecting and recording bugs during the software design and development and after the release. All feedback recorded as bugs shall be evaluated to determine whether a problem exists in the software product, the deliverable, or activities.

3.1.2 | Bug issues

A bug issue can be in one of the statuses reported in the Table 1.

Once the bug has been reported, internally or externally to the organization, a bug issue is created (status OPEN), and tasks and sub-tasks are assigned and performed within the team (status IN PROGRESS). When the software is implemented according to the bug issue, the status

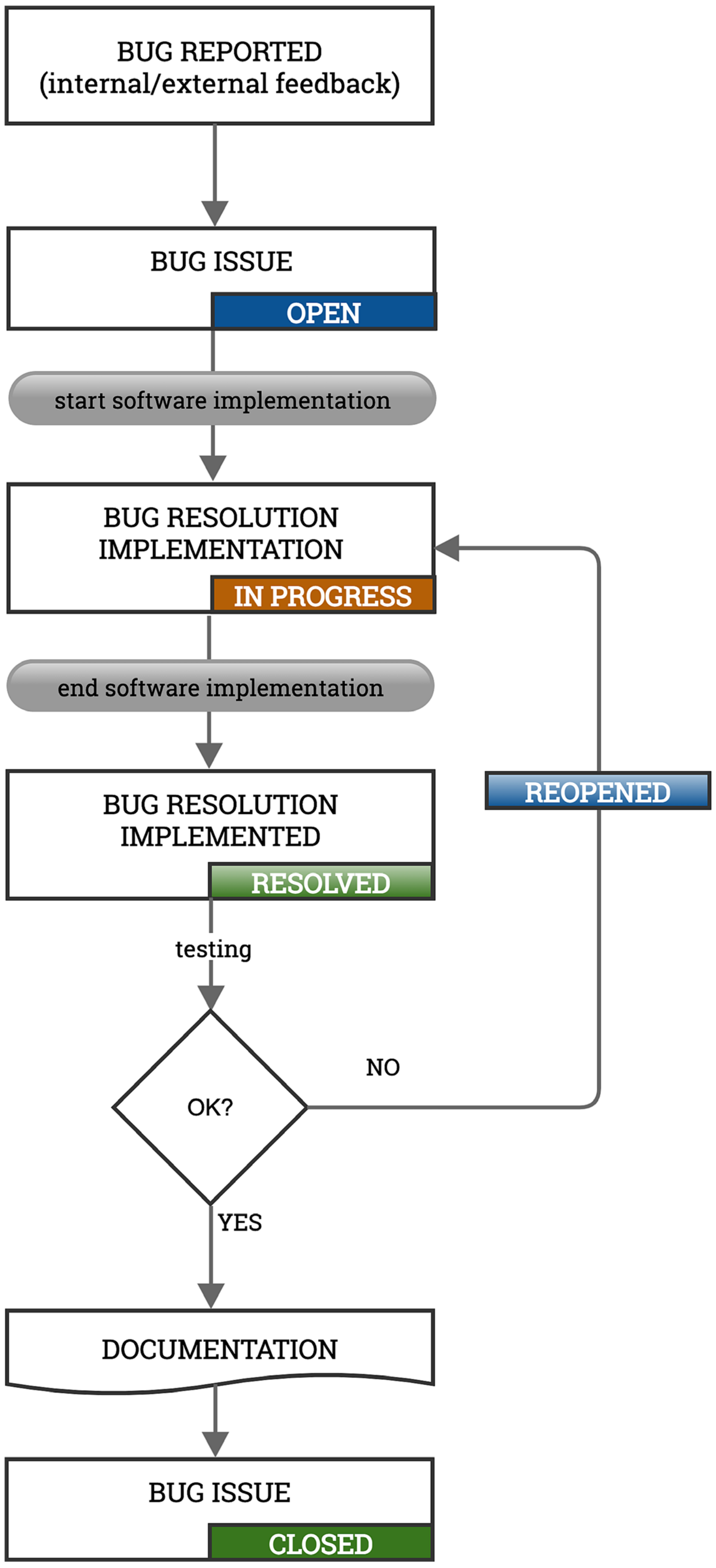

BUG REPORTED
(internal/external feedback)
BUG ISSUE
OPEN
start software implementation
BUG RESOLUTION IMPLEMENTATION
IN PROGRESS
end software implementation
BUG RESOLUTION IMPLEMENTED
RESOLVED
REOPENED
testing
OK?
NO
YES
DOCUMENTATION
BUG ISSUE
CLOSED

FIGURE 5 Flow chart of a Jira bug issue.

changes in RESOLVED. A series of tests are run to verify if the bug is fixed. If yes, the RESOLVED status is complete; however, crucial tasks related to the bug issue are needed before closing it, for example, the creation of documentation. Furthermore, an automatic mechanism has been implemented to not close the bug issue until all tasks are completed. If the tests fail, the bug issue's status changes to REOPENED for implementation (IN PROGRESS), then RESOLVED, and finally CLOSED.

Figure 5 shows a flowchart of the whole bug issue management.

3.1.3 | Bug recording

Once a software problem is collected, a new bug issue in Jira is created.

A bug issue contains the fields reported in the Table 2.

The bug issue can have sub-tasks that can be assigned and tracked individually. A sub-task can be created for an issue either to split the issue into smaller chunks or to allow various aspects of an issue to be assigned to different people. The sub-task can be in one of the following statuses: TO DO, the sub-task has been created; IN PROGRESS, the sub-task is in implementation; and DONE, the sub-task has been implemented. Bug issues and their sub-tasks can have comments and/or attachments that document the activity.

3.1.4 | Investigate the problem

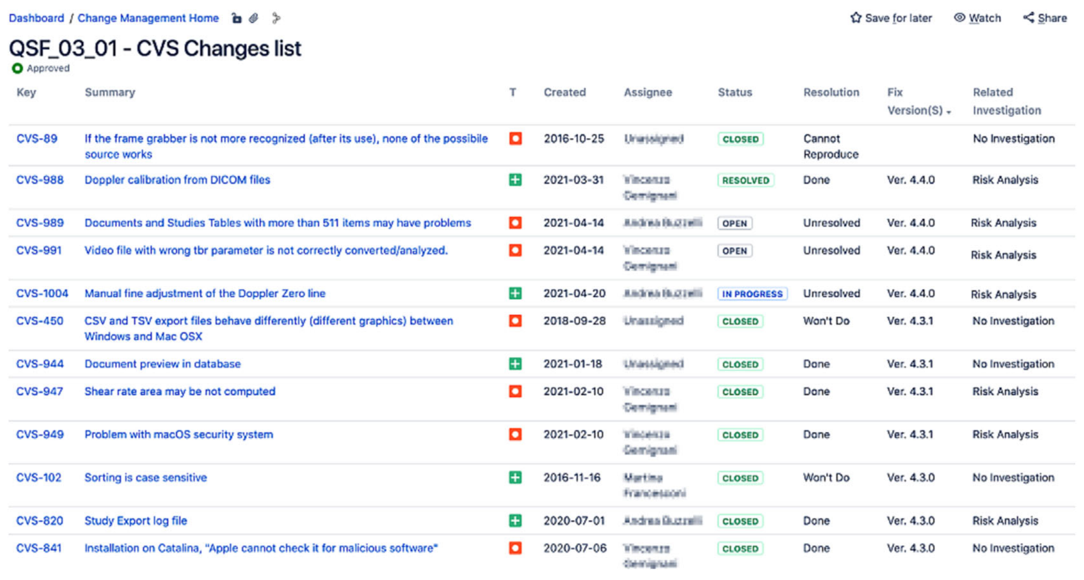
Each bug issue shall be evaluated to determine how it affects the safety and effectiveness of a released software product, according to the internal risk management procedure and which change to the software product is needed to address the problem. The scope of the change is evaluated (software versions affected, resources, and time needed to fix the problem).

The responsible for maintenance process shall investigate the problem, check if it can be replicated, check which versions of the software are affected, identify possible causes and identify a possible software change for the problem resolution. The bug issue is automatically reported in a dedicated "change list" (Figure 6) that is periodically reviewed by the quality manager.

The bug issue is analyzed taking into account its impact on safety and effectiveness, and a decision is taken on the need for a further investigation. The subsequent steps can be managed through a risk analysis, corrective or preventive action, non-conformity, communication to third parties (such as competent authority), or other needed process. The outcome of the investigation (if any) is documented in the "change list." When required by the results of the investigation, the problem shall be managed also according to details reported in the

TABLE 2 Required field and description for each bug issue.

Field	Description
ID	ID of the bug issue
Project	Name of the software project
Summary	Short description of the bug
Description	Detailed description of the bug
Priority	Priority of the bug resolution among highest, high, medium, low, and lowest
Component/s	Software component/s affected by the bug
Fix version	Software version where the bug resolution is implemented
Reporter	Who reported the bug
Assignee	Who is responsible for the implementation of the bug resolution
Environment	Environment (i.e., operating systems) where the bug is present
Affects version/s	Software version where the bug was detected
Status	According to the added workflow
Resolution	Resolved or unresolved
Detected by	Who detected the bug: internal user, external user, development, quality assurance
Detected by notes	Notes regarding who detected the bug and who and when the bug registration was authorized
Related investigation	Issue related investigation (no investigation, risk analysis, CAPA, third parties communication, non-conformity, other)
Related investigation notes	Notes regarding the related investigation



Dashboard / Change Management Home

Save for later · Watch · Share

QSF_03_01 - CVS Changes list

Approved

Key	Summary	T	Created	Assignee	Status	Resolution	Fix Version(S)	Related Investigation
CVS-89	If the frame grabber is not more recognized (after its use), none of the possibile source works	🟥	2016-10-25	[redacted]	CLOSED	Cannot Reproduce		No Investigation
CVS-988	Doppler calibration from DICOM files	🟩	2021-03-31	[redacted]	RESOLVED	Done	Ver. 4.4.0	Risk Analysis
CVS-989	Documents and Studies Tables with more than 511 items may have problems	🟥	2021-04-14	[redacted]	OPEN	Unresolved	Ver. 4.4.0	Risk Analysis
CVS-991	Video file with wrong tbr parameter is not correctly converted/analyzed.	🟥	2021-04-14	[redacted]	OPEN	Unresolved	Ver. 4.4.0	Risk Analysis
CVS-1004	Manual fine adjustment of the Doppler Zero line	🟩	2021-04-20	[redacted]	IN PROGRESS	Unresolved	Ver. 4.4.0	Risk Analysis
CVS-450	CSV and TSV export files behave differently (different graphics) between Windows and Mac OSX	🟥	2018-09-28	[redacted]	CLOSED	Won't Do	Ver. 4.3.1	No Investigation
CVS-944	Document preview in database	🟩	2021-01-18	[redacted]	CLOSED	Done	Ver. 4.3.1	No Investigation
CVS-947	Shear rate area may be not computed	🟥	2021-02-10	[redacted]	CLOSED	Done	Ver. 4.3.1	Risk Analysis
CVS-949	Problem with macOS security system	🟥	2021-02-10	[redacted]	CLOSED	Done	Ver. 4.3.1	Risk Analysis
CVS-102	Sorting is case sensitive	🟩	2016-11-16	[redacted]	CLOSED	Won't Do	Ver. 4.3.0	No Investigation
CVS-820	Study Export log file	🟩	2020-07-01	[redacted]	CLOSED	Done	Ver. 4.3.0	Risk Analysis
CVS-841	Installation on Catalina, "Apple cannot check it for malicious software"	🟥	2020-07-06	[redacted]	CLOSED	Done	Ver. 4.3.0	No Investigation

FIGURE 6 Change list of issues. This in an example of how it is possible to add the *JIRA issue* macro in a Confluence page in order to have a list of issues that is populated automatically. Once you have added the macro, you can customize how the list of issues appears on the page, including how much information to display, how many issues, and so on. It is possible to configure this view through an advanced search that allows to build structured queries using the JIRA Query Language (JQL) to search for issues and to specify criteria that cannot be defined in the quick or basic searches.

internal procedure for post-market surveillance. All relevant parties shall be informed about (i) any problem in released software products and the consequences of continued unchanged use and (ii) the nature of any available changes to released software products and how to obtain and install the changes.

Once the bug has been investigated, a priority is assigned to the bug (among highest, high, medium, low, and lowest) according to the results of the investigation. A version of the software for the implementation of the bug resolution can be also reported. The responsible for maintenance process assigns the issue and in conjunction with the quality manager assign its sub-tasks (if any) to the staff members and move the issue to the IN PROGRESS status, so that the implementation of the bug resolution can start. When required by the results of the investigation, the responsible for maintenance process shall give the highest priority and starts the problem resolution immediately. If required, an ad hoc release of the software shall be programmed, where the bug is resolved.

3.1.5 | Software change implementation for the bug resolution

The software changes needed to solve the bug are implemented by the product design team. The assignee will be the person in charge for the software implementation. Once the change is implemented, it undergoes a dedicated testing phase. For verifying the correct implementation, an ad hoc test can be created and executed; alternatively, it is possible to execute an existing test from those available (the test execution shall be linked to the Jira issue). Once the implemented issue is tested, if the test is successful, the assignee shall move the issue in the RESOLVED status and the resolution field will be DONE. Otherwise, the assignee shall continue working on the issue until he or she obtains the expected result. If the product design team decided not to proceed, the issue can be anyway moved in the RESOLVED status and the resolution field will be WON'T DO.

3.1.6 | Software change approving

Once RESOLVED, if the resolution is DONE, the software change shall be

a. closed and integrated in the new version of the product;
b. properly documented in the technical documentation; and
c. fully tested according to the software test plan.

Otherwise, the issue can be moved to the CLOSED status or to the REOPENED status according to its final evaluation by the product design team. If REOPENED, the software change shall be accepted with the same process of the OPEN issues mentioned above.

3.1.7 | Change request not deriving by a problem resolution process

Inputs for software changes may arise not only by a problem resolution need and may come from anyone in the company, from external consultants, subcontractors, and from customers. These inputs shall be reported to the responsible for maintenance process who can initialize the design and process change request by creating a "software change" issue in SWMA, similarly to the bug issues mentioned in the previous paragraphs. Software change issues are intended for implementing specific and limited changes aimed to take improvements to the original project in fields

TABLE 3 IEC 62304 requirements related to the problem resolution process.

IEC 62304 clause mentioning the problem resolution process	Specific points	Compliance of the implemented architecture	Software risk class (A, B, C).
5.1 Software development planning	5.1.1 Software development plan 5.1.9 Software configuration management planning	All these activities are managed thanks to a quality management system (ISO 13485) including planning and maintenance processes. A responsible is identified for handling problems at each stage of the software life cycle. All feedback recorded are evaluated to determine whether a problem exists in the software product, the deliverable or activities. The software configuration management information includes that once a software problem is collected, a new bug issue in JIRA is created.	A, B, C
5.6. Software integration and integration testing	5.6.8 Use software problem resolution PROCESS	The software problem resolution process based on JIRA is applied to the anomalies found during software system, integration, and integration testing. The bug issue is reported in a dedicated "change list" that is periodically reviewed by the quality manager.	B, C
5.7 SOFTWARE SYSTEM testing	5.7.2 Use software problem resolution PROCESS		B, C
6 Software maintenance process	6.1 Establish software maintenance plan 6.2.2 Use software problem resolution PROCESS	The planned and implemented software problem resolution process analyzes and solves, through the implemented digital solution, problems arising after the release of the medical device software addressing problem reports.	A, B, C
9 Software problem resolution process	9.1 Prepare PROBLEM REPORTS 9.2 Investigate the problem 9.3 Advise relevant parties 9.4 Use change control process 9.5 Maintain records 9.6 Analyze problems for trends 9.7 Verify software problem resolution 9.8 Test documentation contents.	Problems detected from different sources and affecting software are documented as "bug" issues in JIRA. The responsible for maintenance process investigates the problem and identify possible causes and a possible software change for the problem resolution. Each bug issue is evaluated to determine how it affects the safety and effectiveness of a released software product, according to the internal risk management procedure. Bug issues are created for actions needed to correct the problem, or document the rationale for taking no action. All relevant parties are informed about the existence of the problem in released software products and the nature of any available changes implemented observing the requirements of the change control process. Records of problems and their resolution are appropriately documented including their verification. The bug issue assigns sub-tasks that can be tracked individually and allows to verify if the problem has been resolved and the problem report has been closed, adverse trends have been reversed, change requests have been implemented in the appropriate software products and activities, and additional problems have been introduced. Documentation containing test results, anomalies found, the version of software tested, relevant hardware and software test configurations, relevant test tools, date tested, and identification of the tester is always tracked and available.	A, B, C

TABLE 4 IEC 62304 requirements related to the change request management.

IEC 62304 clause mentioning the change request process	Specific points	Compliance of the implemented architecture	Software risk class (A, B, C).
6.2 Problem and modification analysis	6.2.3 Analyze CHANGE REQUESTS 6.2.4 CHANGE REQUEST approval 6.2.5 Communicate to users and regulators	Each change request is reported to the responsible for maintenance process who initializes the design and process change request by creating a "software change" issue. Change effect on the organization, released software products, and systems are analyzed. The change request is recorded and, if approved, resulting changes can be managed trough sub-tasks that can be assigned and tracked individually. The approved change requests that affect released software products are identified, so users and regulators are informed about any consequences of continued unchanged use, the nature of any available changes and how to obtain and install the changes.	B, C (6.2.3) A, B, C (6.2.4-6.2.5)
6.3 Modification implementation	6.3.1 Use established PROCESS to implement modification 6.3.2 Re-release modified SOFTWARE SYSTEM	Software development and maintenance processes are established and used, thanks to the developed architecture, to implement the modifications. Modifications may be released as part of a full re-release of a software system or as a modification kit comprising changed software items and the necessary tools to install the changes as modifications to an existing software system.	A, B, C
8.2 Change control	8.2.1 Approve CHANGE REQUESTS 8.2.2 Implement changes 8.2.3 Verify changes 8.2.4 Provide means for TRACEABILITY of change	The change is implemented as specified in the change request and undergoes a testing phase and any activity and verification that needs to be repeated as a result of the change, including changes to the software safety classification of software systems and software items. Each change request, relevant problem report, and approval of the change request is ever traced and available thanks to the developed architecture.	A, B, C
9 Software problem resolution process	9.4 Use change control process	All approved and implemented change requests are in accordance with requirements of the change control process.	A, B, C

such as reliability, architecture, usability, serviceability, or manufacturability. More comprehensive changes requiring a strong revision of the original project might also require to be evaluated according to the internal software design and development procedure.

The change request is recorded, and then, if approved, resulting modifications are managed as "software change" issues in Jira.

The software change issue status flow follows the steps of the bug issue reported in Figure 3 and contains the same fields. Similarly, the software change issue can be managed through sub-tasks that can be assigned and tracked individually.

3.2 | Requirements' fulfillment

The requirements described in the previous paragraph are mentioned in Clauses 5, 6, 8, and 9 of 62304. Tables 3 and 4 show how the specific points have been addressed, by the implemented digital solution, for problem resolution and change control, respectively.

It is worth noting that the group has a quality management system that successfully underwent external audits (years 2018-2019-2020-2021-2022) according to UNI EN ISO 13485: 2016 international standard and the 93/42/EEC Medical Device Directive (MDD). Moreover, the certification for conformity to the new EU MDR has been recently obtained for the developed software medical device (December 2022).

4 | CONCLUSIONS

The growing number of regulatory requirements that medical devices must comply with leads to the need for automated and digital approaches to better manage products' development and maintenance processes throughout their life cycle.

In particular, software medical devices are highly complex and dynamic systems that are affected by a large number of changes and implementations throughout their life cycle. Each modifications can generate unwanted effects, so it is necessary to verify and test each change also in relation to the risk analysis and the impact on the entire system.[12]

It is mandatory that MDSWs provide safety, reliability, and security because failure to comply with these aspects can lead to injury or be fatal. Manufacturers who can rely on an efficient quality management system can implement procedures that simplify compliance with needed requirements and the traceability of relevant processes as risk management, problem resolution and change management. However, the control, management, and traceability of the development and maintenance processes can be difficult and expensive tasks.[23]

To facilitate access to the market of innovative medical device software while ensuring high-level for safety and health, it is necessary to digitize processes and to use tools for interconnection between the people of the company. In this scenario, we proposed a cost-effective and fully customizable solution for problems' and modifications' management, based on a low-cost commercial platform and implemented in our group, the developed approach can help manufacturers/groups to navigate the complex regulatory network and to manage life cycle processes for the safe development and maintenance of a medical device software.

In particular, the architecture solution, based on the commercial tool, was described addressing reference medical devices requirements both in terms of problem resolution and change control representing crucial aspects for performance, safety, or regulatory clearance of a medical device as mentioned in the IEC 62304 standard.

Our solution presents high degree of customization and configurability that can be exploited inside an organization by internal expertise and know-how. Furthermore, our solution allows the needed integration of documentation control with implementation, versioning, modification processes, and compliance with standards throughout the whole life cycle of the product. In addition, the implemented approach allows to manage projects across a team from planning to execution with organized workflows, task and access tracking, and integrated documentation management.

It is worth noting that the presented architecture can be useful for manufacturers to establish software maintenance plans that include activities and tasks for analyzing and resolving problems arising before or after the release of the medical device software.

Efficient solutions that are economically and operatively sustainable are needed. Control cannot become more challenging than development itself, it must be a smart and agile tool to aid manufacturers and developers in ensuring effectiveness of the medical device and safety of users.

In this scenario, our approach is based on the management of the maintenance processes of MDSWs in favor of quality and safety in an efficient and digital way to simplify procedures and facilitate control by the regulatory bodies, guaranteeing reliability through compliance to standards.

CONFLICT OF INTEREST STATEMENT

EB and VG are co-founders of QUIPU s.r.l., Pisa, Italy, a spin-off company of the Italian National Research Council and the University of Pisa developing software medical devices.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available from the corresponding author upon reasonable request.

ORCID

Maria Raffaella Martina https://orcid.org/0000-0003-2489-8639

REFERENCES

1. MDCG. *Guidance on Qualification and Classification of Software in Regulation (EU) 2017/745 – MDR and Regulation (EU) 2017/746 – IVDR*. Published online; 2019:1-28. https://ec.europa.eu/docsroom/documents/37581/attachments/1/translations/en/renditions/native
2. Blagec K, Jungwirth D, Haluza D, Samwald M. Effects of medical device regulations on the development of stand-alone medical software: a pilot study. *Stud Health Technol Inform*. 2018;248:180-187. doi:10.3233/978-1-61499-858-7-180
3. Jarow JP, Baxley JH. Medical devices: US medical device regulation. *Urol Oncol Semin Orig Investig*. 2015;33(3):128-132. doi:10.1016/j.urolonc.2014.10.004
4. *MDR Delay Official as Industry Calls to Push Back IVDR|RAPS*. Accessed April 23, 2021. https://www.raps.org/news-and-articles/news-articles/2020/4/mdr-delay-moves-forward-as-industry-calls-to-push
5. Council Directive. *93/42/EEC of 14 June 1993 Concerning Medical Devices*. Published online 1993.
6. McCarthy AD, Lawford PV. Standalone medical device software: the evolving regulatory framework. *J Med Eng Technol*. 2015;39(7):441-447. doi:10.3109/03091902.2015.1088087
7. Bhatia P, Ali LCC, Burri KG, Pritchard G, Singh N, Upadhyay N. New documents required by the medical device regulation. *Med Writ*. 2020;29(3):24-29.
8. Becker K, Lipprandt M, Röhrig R, Neumuth T. Digital health - software as a medical device in focus of the medical device regulation (MDR). *IT - Inf Technol*. 2019;61(5-6):211-218. doi:10.1515/itit-2019-0026
9. BSI. *Medical Device Software — Software Life-Cycle Processes*. Bs En *623042006 +a12015*. Vol. 3(November 2008); 2015. 88p. https://www.iso.org/standard/64686.html

10. Värri A, Kranz-Zuppan P, De La Cruz R. IEC 62304 ed. 2: software life cycle standard for health software. *Stud Health Technol Inform*. 2019;264:868-872. doi:10.3233/SHTI190347
11. Bianchini E, Francesconi M, Testa M, Tanase M, Gemignani V. Unique device identification and traceability for medical software: a major challenge for manufacturers in an ever-evolving marketplace. *J Biomed Inform*. 2019;93:103150. doi:10.1016/j.jbi.2019.103150
12. Hrgarek N. Certification and regulatory challenges in medical device software development. In: *2012 4th International Workshop on Software Engineering in Health Care (SEHC)*; 2012:40-43. doi:10.1109/SEHC.2012.6227011
13. Laukkarinen T, Kuusinen K, Mikkonen T. DevOps in regulated software development: Case medical devices. In: *Proc - 2017 IEEE/ACM 39th Int Conf Softw Eng New Ideas Emerg Results Track, ICSE-NIER 2017*. Published online; 2017:15-18. doi:10.1109/ICSE-NIER.2017.20
14. The 2019 Accelerate State of DevOps: Elite performance, productivity, and scaling|Google Cloud Blog. Accessed February 1, 2023. https://cloud.google.com/blog/products/devops-sre/the-2019-accelerate-state-of-devops-elite-performance-productivity-and-scaling
15. Jabbari R, bin Ali N, Petersen K, Tanveer B. Towards a benefits dependency network for DevOps based on a systematic literature review. *J Softw Evol Process*. 2018;30(11):e1957. doi:10.1002/smr.1957
16. Lwakatare LE, Karvonen T, Sauvola T, et al. Towards DevOps in the embedded systems domain: why is it so hard? In: *Proc Annu Hawaii Int Conf Syst Sci*. Vol.2016-March; 2016:5437-5446. doi:10.1109/HICSS.2016.671
17. Erich FMA, Amrit C, Daneva M. A qualitative study of DevOps usage in practice. *J Softw Evol Process*. 2017;29(6):1-20. doi:10.1002/smr.1885
18. Angel M, Bermejo P. *DevOps: Is There a Gap Between Education and Industry?* 2023;(October 2022):1-25. doi:10.1002/smr.2534
19. *Half of All DevOps Teams Can't Measure Performance - Work Life by Atlassian*. Accessed April 23, 2021. https://www.atlassian.com/blog/devops/new-devops-metrics?utm_source=website&utm_medium=link&utm_campaign=devops_moment_2_launch
20. Atlassian. *DevOps Trends Survey*. Published online. Vol. 2020; 2020:1-38. https://www.atlassian.com/whitepapers/devops-survey-2020
21. FDA. *Content of Premarket Submissions for Management of Cybersecurity in Medical Devices*. FDA Guid. Published online; 2018:6. https://www.fda.gov/media/119933/download
22. Medical Device Coordination Group. *Guidance on Cybersecurity for Medical Devices*. Mdcg 2019-16. Published online; 2019:1-46. https://ec.europa.eu/docsroom/documents/38941
23. Regan G, Biro M, Mc Caffery F, Mc Daid K, Flood D. A traceability process assessment model for the medical device domain. In: *Communications in Computer and Information Science*. Vol.425. Springer Verlag; 2014:206-216. doi:10.1007/978-3-662-43896-1_18

How to cite this article: Martina MR, Bianchini E, Sinceri S, Francesconi M, Gemignani V. Software medical device maintenance: DevOps based approach for problem and modification management. *J Softw Evol Proc*. 2024;36(4):e2570. doi:10.1002/smr.2570