Guidelines and Recommendations

Hanneke W.M. van Deutekom and Saskia Haitjema*

Recommendations for IVDR compliant in-house software development in clinical practice: a how-to paper with three use cases

https://doi.org/10.1515/cclm-2022-0278
Received March 22, 2022; accepted April 26, 2022;
published online May 11, 2022

Abstract

Objectives: The *In Vitro* Diagnostics Regulation (IVDR) will be effective in May 2022 by which in-house developed tests need to apply to the general safety and performance requirements defined in Annex I of the IVDR ruling. Yet, article 16 from Annex I about software can be hard to interpret and implement, particularly as laboratories are unfamiliar with quality standards for software development.
Methods: In this paper we provide recommendations on organizational structure, standards to use, and documentation, for IVDR compliant in-house software development.
Results: A practical insight is offered into novel standard operating procedures using three examples: an Excel file with a formula to calculate the pharmacokinetics of tacrolimus and to calculate the new dose, a rule for automated diagnosis of acute kidney injury and a bioinformatics pipeline for DNA variant calling.
Conclusions: We recommend multidisciplinary development teams supported by higher management, use of ISO-15189 in synergy with IEC-62304, and concise documentation that includes intended purpose, classification, requirement management, risk management, verification and validation, configuration management and references to clinical or performance evidence.

Keywords: *in vitro* diagnostics regulation (IVDR); medical device software; quality management; software development.

***Corresponding author: Saskia Haitjema**, Central Diagnostic Laboratory, UMC Utrecht, Utrecht University, Utrecht, the Netherlands, Phone: +31 88 75 57155,
E-mail: s.haitjema@umcutrecht.nl. https://orcid.org/0000-0001-5465-4868
Hanneke W.M. van Deutekom, Department of Genetics, UMC Utrecht, Utrecht University, Utrecht, the Netherlands. https://orcid.org/0000-0002-2429-1098

Introduction

The right treatment starts with the right diagnosis. Patients and healthcare professionals therefore need to be able to trust medical devices that help them make a reliable diagnosis, such as *in vitro* diagnostic medical devices. With the protection of health for patients and users of *in vitro* diagnostic medical devices in mind, the European Union (EU) recently updated its standards for quality and safety for *in vitro* diagnostic medical devices [1]. The *In Vitro* Diagnostics Regulation (IVDR) will be effective in May 2022 by which in-house developed tests need to apply to the general safety and performance requirements defined in Annex I of the IVDR ruling [1, 2].

Compared with the *In Vitro* Diagnostic Directive (IVDD), two significant changes impact current laboratory practice. First, IVDR now specifically refers to laboratory software, including in-house developed software [1]. Furthermore, the IVDR includes specific conditions for in-house developed devices. These conditions state that healthcare institutions can still develop their own tests without the need for CE marking [1].

Currently, laboratories are investigating their practices regarding software to comply with IVDR. Article 16.2 of the IVDR states that “software shall be developed and manufactured in accordance with the state-of-the-art taking into account the principles of development life cycle, risk management, … ” [1]. This article can be hard to interpret and implement, particularly as laboratories are unfamiliar with quality standards for software development. This requires a multidisciplinary collaborative effort, as bioinformaticians and developers of digital health products enter the 21st century laboratory environment that used to be dominated by (wet-)laboratory specialists. Especially in highly specialized clinical facilities that already harbor such innovative teams, questions regarding the opportunities and limitations of IVDR may arise.

At the UMC Utrecht division of Laboratories, Pharmacy and Biomedical Genetics, we have recently aligned IEC-62304 for medical software development with the

Open Access. © 2022 Hanneke W.M. van Deutekom and Saskia Haitjema, published by De Gruyter. This work is licensed under the Creative Commons Attribution 4.0 International License.

ISO-15189 medical laboratory quality management system to be able to leverage the potential of article 5.5 and create in-house developed software in an IVDR compliant manner [1]. The IEC-62304 is a harmonized standard that covers safe design, development and maintenance of medical software. In this paper we provide recommendations for IVDR compliant in-house developed software and a practical insight into our novel standard operating procedures using three examples.

What is IVDR software?

In article 2 of IVDR, software is specifically mentioned in the formal definition of an *in vitro* diagnostic medical device. Briefly, software is considered IVDR software if it is intended to be used (alone or in combination) for the examination of specimens derived from the human body to provide information on physiological or pathological processes, predisposition of diseases and/or to predict treatment responses [1]. IVDR software can often be seen as software that will drive or inform clinical management, i.e. the results from the software are used to diagnose a patient or to provide information about which treatment to start.

Several guidelines have been created by the Medical Device Coordination Group (MDCG) to help interpret the IVDR. The infographic "Is your software a Medical Device?" uses decision steps to assist in qualification of IVDR software [3]. Of particular interest is the guidance on qualification and classification of software (MDCG2019-11 [4]). This guidance states that a laboratory information system is not considered IVDR software as it is merely a storage system. Any addition to the system that performs calculations that exceed basic operations (e.g. transpose US to EU metrics) may be considered an IVDR software module. Websites containing risk calculators and Excel sheets with fixed or flexible formulas are thus subject to the IVDR. Noteworthy, Excel is widely used and provides an intuitive and easy to use interface for (laboratory) specialists. Within our division >50 Excel files contain formulas that generate information on which a diagnosis can be made or which creates/alters a treatment plan, and are thus considered IVDR software. Moreover, with the arrival of big data within the clinic, more complex examples have arrived, where specialized software and algorithms, developed by bioinformaticians and data scientists have become an essential part of diagnostic laboratories.

If no equivalent CE-labeled software product is available for the specific requirements of the laboratory, health institutions that are established in the EU are allowed to develop their own IVDR software if they meet specific conditions and comply with the general safety and performance requirements in the IVDR detailed in article 5.5 and Annex I [1]. For example, in-house developed software cannot be transferred to other legal entities and needs to be manufactured under an 'appropriate quality management system'.

Below, we share recommendations on how to meet IVDR requirements for software in a laboratory environment, synergizing good laboratory practices and quality management systems that already are part of our day-to-day work.

Recommendations for organizational structure

Successful innovation teams are able to bridge the gap between research and clinical practice efficiently. Rather than appointing specific innovation officers, health institutions can start from existing quality-aware personnel, by freeing up time from different stakeholders to participate in multidisciplinary development teams. In our hospital, software engineers, laboratory specialists and clinical specialists work together to translate diagnostic questions to specific requirements for software and clinically relevant intended purposes. Depending on the software that is being built, 'software engineers' in this case includes laboratory specialists (for formulas in Excel), laboratory information system administrators (for modules or simple rules) and/or bioinformaticians (for complex pipelines). Either way, developing software works best in an agile way, where improvement cycles build towards a minimal viable product so end users can co-steer the development based on early results. Embedding multidisciplinary teams within clinical care further facilitates a short quality feedback loop after the developed IVDR software is put into service. Formal higher management support for this approach, including compliance to appropriate standards, is of paramount importance, as responsibility for in-house development and its quality is at a health institution level.

Recommendations for standards to use

Specific standards

In article 5.5(c) the IVDR refers to the ISO-15189, requirements for quality and competence of medical laboratories, and additional national provisions. However in Annex I article 16, on life cycle processes or software development, IVDR does not specifically state which standard to use: "the software

shall be developed and manufactured in accordance with the state of the art […].” [1]. In addition to the ISO-15189 we recommend using the IEC-62304 for medical device software as compliance to this harmonized standard guarantees adequate documentation of software.

In-house developed software needs to comply with the complete list in Annex I. To help interpreting Annex I from an in-house developed software perspective, we have extended the list of the Dutch IVDR Task Force [5]. This list now links Annex I of the IVDR to both sections of the ISO-15189:2012 and sections of IEC-62304 (Supplementary Material 1). The list provided by the Dutch IVDR Taskforce applies to in-house developed software only, i.e. this addition is not applicable to websites or health apps [1].

Recommendations for documentation

Life cycle process

The IEC-62304 is not mandatory, and for simple software modules or simple Excel sheets, following all sections in this standard might be too much. We recommend documentation that minimally includes: (i) Intended purpose; (ii) Classification; (iii) Requirement management; (iv) Risk analysis/management; (v) Verification and validation; (vi) Configuration management; (vii) Reference to clinical or performance evidence. When accredited for ISO-15189 the following items are most likely already covered: (viii) Release date; (ix) Process owner, verifier, authorizer; (x) Incidents, problems and control measures; and (xi) Post Market Surveillance.

Risk classification

We recommend using MDCG 2019-11 [4] to see whether your software should comply with the IVDR and MDCG 2020-16 [6] for the classification. In addition to the IVDR classification A-D, IEC-62304 has its own classification system, class A-C. To avoid confusion, IVDR or IEC-62304 should be indicated when mentioning any classification.

The classification of the IEC-62304 considers the risks of failure due to the software, and ranges from “failure of the software does not contribute to a dangerous situation” (class A) to “failure of the software results in death or serious injury” (class C). External risk control measures may reduce the probability that software failure causes harm, and these control measures can be taken into account for the IEC-62304 classification. In general, for IVDR software, nobody will die or be seriously injured *because* of the failure of IVDR software. However, this does not imply there is no harm, as people might die or be seriously injured as a result of incorrect clinical management caused by an incorrect result of IVDR software. The main risks due to failure of IVDR software are (1) Incorrect result; (2) Delayed result; and (3) No result. Whether a delayed result is acceptable depends on the requirements of the end user. In most cases, information gained from IVDR software will not directly influence a patient as diagnostic information will be interpreted by at least one specialist, which significantly mitigates the risk. Results indeed “drive clinical management” or “inform clinical management”, which is different from “to treat” or “to diagnose”.

Requirement management

Software design should be based on both requirements and risk analysis. Requirements need to be properly defined prior to the start of the development process. Additionally, as the IVDR requests a marketscan, the initial requirements can be used to select or reject existing CE-IVD certified software products. We recommend the IMDRF (International Medical Device Regulators Forum) document about ”Possible Framework for Risk Categorization and Corresponding Considerations” for different aspects of requirements to set for software [7].

New functionalities, user experience, market developments and new techniques can lead to new requirements for the software. Hence, it is recommended to check and update requirements with each release.

Risk management

A risk analysis must be documented at the start of the software development process. For high risks, control measures are needed that are demonstrably secured, and the residual risks must be accepted by qualified personnel. The risk analysis is further expanded during the design phases of software, and through user experience. Recently, ISO-22367 appeared; “Medical laboratories – Application of risk management to medical laboratories”, which is linked to ISO-15189. Annex D contains – among others – specific software questions. Inspiration can also be found in the above-mentioned IMDRF document [7].

Verification and validation

In software development, verification confirms that the code works for the specific requirements of the code and

can include unit tests and integration tests. Validation confirms that the software fulfills its intended purpose; e.g. whether the needs of the end users are met. The "V-model" [8], a well-known model for verification and validation of software, may provide inspiration. The IEC-62304 requires specific sections to be part of the Validation procedure – among others – the test itself, expected results, test results, evaluation and conclusion. Noteworthy, the definition of verification and validation differ with those definitions of the ISO-15189. We recommend using the IEC-62304 definitions for software development.

Configuration management

Software development is a continuous process and thus requires versioning. The version of a software product is often displayed as several numbers delimited by a dot. The first number indicates the major number, and only changes with large upgrades of the software. The second number is the minor number and increases when new functionalities and/or bug fixes are released. The third number is often used for a hotfix; with a specific release to fix a crucial bug. The fourth number is the build number, these development builds are created throughout the life cycle and used for integration testing. For in-house developed software, it is recommended to define and document which numbers are used, and when to change them.

Post marketing surveillance (PMS)

The idea behind PMS is to ensure the real-life safety of medical devices. For in-house developed tests, particularly when the laboratory is accredited with ISO-15189, PMS is already implemented. In particular article 4.14.1, 4.14.3, 4.14.3 and 4.14.7, which include assessing user feedback, allow for suggestions for improvements, and add quality indicators to monitor performance among all processes. We recommend to include quality indicators for the performance of in house developed software.

Reference to clinical or performance evidence

We recommend looking into Guidance 2020-01 on Clinical Evaluation (MDR)/Performance Evaluation (IVDR) of Medical Device Software from the MDCG [9].

Software development

ISO-15189, art. 5.5.3, states that procedures should be documented. Software development is a procedure on its own. Environments like Github or Bitbucket are often used for software development and allow for version control systems to track the source code changes during software development. How these environments are used should be properly documented. This documentation should include release procedures, including hotfixes, verification and validation procedures, requirement management and risk management.

Using off-the-shelf software

One may wonder whether it's allowed to incorporate tools or off-the-shelf software in your in-house developed software product. In the IEC-62304, these are referred to as SOUPs; software of unknown provenance, and special considerations are provided. Examples of SOUPs are BWA [10] which is used for mapping DNA sequences to a reference genome, and GATK [11] used for DNA variant calling. One is required to at least document the intended purpose of the SOUP and whether it is used for a different purpose. Requirements and validation testing for the specific use of the SOUP should be included in the documentation. Furthermore, investigating the risks of using this SOUP is recommended, i.e. what if a new version is not compatible, or what if this software is no longer available.

Use case 1: pharmacokinetics of tacrolimus

There is a large intra- and interindividual variability in the pharmacokinetics for tacrolimus so monitoring of tacrolimus blood concentrations, with multiple sampling moments after intake, is useful for therapeutic drug monitoring. With multiple samples the systemic exposure can be predicted by calculating the area under the curve (AUC) using the trapezoidal rule [12, 13]. The hospital pharmacy uses an Excel file to calculate the pharmacokinetics of tacrolimus using the trapezoidal rule for the AUC and to calculate the new dose. As monitoring of levels of medicinal products is considered class C in the IVDR, this Excel file is considered IVDR software. When implementing the IVDR in the UMC Utrecht, this simple Excel file was one of the first we encountered and its revised version served as a template for the whole division. The revised

version consists of two additional sheets for QMS-related documentation (Supplementary Material 2). The documentation sheet includes (i) Intended purpose of the module; (ii) risk classification; (iii) requirements; (iv) risk analysis; and (v) Reference to the evidence of clinical performance, which includes a national guideline and two references. The validation sheet includes (i) several tests, which must be repeated with each new version of the Excel file; (ii) evaluation; and (iii) conclusion. The test includes time stamps and concentrations with a verified outcome. One could debate whether testing formulas in Excel is verification or validation. We recommend validation, as the requirements of the end users are tested. Working in an ISO-15189 accredited laboratory, the Quality Management System (QMS) of the lab handles the versioning of this Excel file itself, including the date. Even though we do not expect this Excel file to change in the near future, if it does, all its content will be reconsidered and updated for a version 2. The QMS also handles feedback.

Use case 2: automated diagnosis of acute kidney injury

Acute kidney injury (AKI) is associated with high morbidity and mortality [14] and although its definition is based on simple differences in serum creatinine levels and urine output [15], AKI is often missed. In collaboration with a multidisciplinary team with nephrologists, internists, laboratory specialists and data scientists, we recently implemented a rule-based algorithm for AKI in our Laboratory Information System (LIS), with the outcome visible in the electronic patient record (EHR). One document, containing the complete QMS related documentation, has been created for this module, which includes the following sections (outline in Supplementary Material 3): (i) Intended purpose of the module; (ii) risk classification; (iii) requirements, including a decision tree leading to a yes or no for showing the alert; (iv) risk analysis, including false positive and false negatives outcome and risks for end users; (v) verification of the module; interpreted as an integration test; i.e. does the code provide results; (vi) validation; do the results comply with the end user requests. Specific patients have been selected in order to follow all routes through the decision tree. In the test, patient identifiers for these patients are mentioned and followed in the different steps through LIS and EHR. The acceptance criteria are referring to the decision tree; which should work as requested. The results of the tests came back perfectly and no anomalies were detected (evaluation of the test results). For validation, the people creating and performing the tests have been documented. (vii) Reference to the evidence of clinical performance, which includes 11 references. In summary, it was stated in the document that the AKI alert could be used for diagnostic purposes. The document was verified and authorized by qualified personnel. Similar to use case 1, the QMS of the lab handles the versioning, verification and authorization of the document itself.

Use case 3: bioinformatics pipeline for variant calling in sequencing data

Bioinformaticians have developed a pipeline for variant calling in genetic sequencing data. A pipeline consists of a series of different tools and functions arranged such that the output of each consecutive element is the input of the next. IVDR classifies the pipeline as class C as it is part of genetic testing, and we classified the pipeline as class A following the flowchart in the IEC-62304. Class B of the IEC-62304 seems to be the correct classification, as there is no risk of death when the variant calling is incorrect. Incorrect variant calling could lead to incorrect interpretation which could subsequently lead to incorrect treatment. IEC-62304, however, allows the use of risk mitigation e.g. control measures to be taken into account, which we implemented as follows (1): quality checks throughout the workflow outside of the bioinformatics pipeline that indicate whether or not the data can be trusted and (2) the results of the bioinformatics pipeline are interpreted by a laboratory specialist before release to a geneticist. Thus unexpected results, i.e. a significant change in the number of variants after filtering, will be noticed and the final classification is A.

In our software development procedure, the documentation for each release is combined in an Excel file, where we use a separate sheet for an overview, risk analysis table, requirements, configuration management, SOUPs, related procedures etc. The overview provides the intended purpose, the classification, where the software is installed, links to verification information (on Github) and validation document, information of the last release, such as version, date, commit number, link(s) to Github. For each update of the software all the sheets are updated, to keep the documentation of the software up to date. For software development we use Github, a Git repository hosting service which allows us to track code

changes but also to request code reviews, and track comments about these reviews; which is part of our verification procedure. We track issues using Azure DevOps, which we also use for planning. We have a general email address where we get questions and requests for new features, which end up as an issue, and there are multidisciplinary meetings where bioinformaticians meet the lab personnel, where feedback can be gathered. Genome in a Bottle is used for clinical evidence and performance evaluation [16]. Finally, to validate the results of our pipeline, a comparison is made with results from previous versions of the pipeline, or results from other techniques.

Discussion

Being part of several national working groups of the IVDR and software, we have noticed that there is a gap of knowledge in medical laboratories about how to deal with in-house developed software, using ISO-15189 and IEC-62304 in synergy. Indeed, software development is not incorporated in the ISO-15189, and even in a preview of the ISO-15189:2022 there is no information about documentation for in-house developed software. This paper provides recommendations about what to document, and where to find more information regarding documentation of software as a medical device.

The use cases described in this paper definitely do not represent all IVDR software, however they span across the software spectrum from a calculation in Excel and a simple module to a bioinformatics pipeline. The MDCG 2019-11 [4] is a very good guidance document on what software is considered a medical device. One example that is explicitly mentioned in this guidance is software which is intended to determine the Human HbA_{1c} concentration in serum from the results obtained with a Human HbA_{1c} ELISA. Even though it is considered a trivial task to create a calibration line in, for example Excel, and receive results for diagnostic purposes, it is considered software as a medical device and should be treated and documented as such. Furthermore this guidance states: "Software intended to modify the representation of available *in vitro* diagnostic medical device results is not considered an *in vitro* diagnostic medical device, e.g. basic operations of arithmetic (e.g. mean, conversion of units) and/or plotting of results in function of time, and/or a comparison of the result to the limits of acceptance set by the user." Whether or not the formula, in for example Excel, is considered a basic operation is up to the developer and laboratory specialist. We highly recommend documenting the reasoning when the formula is considered a basic operation.

With IVDR article 5.5a, it is no longer allowed to transfer the devices to another legal entity. This provides a dilemma. In-house developed software is often used for both healthcare and scientific research. In the last few years it has become increasingly important to publish papers with open source code, to be able to contribute to the open science movement and to comply with FAIR principles for software [17].

To comply with IVDR, there should be distinguished steps, such as a stable IT-infrastructure, inclusion in a QMS, and proper validation, between software used for research and the one that is used in a diagnostic setting. In our opinion, this is not a sustainable nor an efficient way of working. In the upcoming years, competent authorities and health institutions should work together in finding creative solutions to this issue that facilitate innovation while honoring the principles of the IVDR.

Conclusions

IVDR is going to significantly change in-house development of IVDR software. Complying with IVDR through creating synergy between ISO-15189 and IEC-62304 is complicated, but it will create value for diagnostics, through multidisciplinary teamwork and careful documentation.

Acknowledgments: We thank Erin Smeijsters and Matthijs van Luin for providing an Excel example and filling in the pharmacology details. We thank Kitty Siemerink, Dörte Hamann, Alexis Kotte, and Daoud Sie for helpful discussions and carefully reading the manuscript.
Research funding: None declared.
Author contributions: HvD and SH conceived the study and wrote the manuscript. All authors have accepted responsibility for the entire content of this manuscript and approved its submission.
Competing interests: Authors state no conflict of interest.
Informed consent: Not applicable.
Ethical approval: Not applicable.

References

1. European Union. In vitro diagnostic regulation; 2017 [Internet]. Available from: https://eur-lex.europa.eu/eli/reg/2017/746/oj.
2. European Commission. Progressive roll-out of the in vitro diagnostic medical devices regulation; 2021 [Internet]. Available

from: https://ec.europa.eu/commission/presscorner/detail/en/ip_21_6965.
3. Medical Device Coordination Group. Is your software a medical device; 2021 [Internet]. Available from: https://ec.europa.eu/health/system/files/2021-03/md_mdcg_2021_mdsw_en_0.pdf.
4. Medical Device Coordination Group. Guidance on qualification and classification of software in regulation (EU) 2017/745 – MDR and regulation (EU) 2017/746 – IVDR; 2019 [Internet]. Available from: https://ec.europa.eu/health/system/files/2020-09/md_mdcg_2019_11_guidance_qualification_classification_software_en_0.pdf.
5. Bank PCD, Jacobs LHJ, van den Berg SAA, van Deutekom HWM, Hamann D, Molenkamp R, et al. The end of the laboratory developed test as we know it? Recommendations from a national multidisciplinary taskforce of laboratory specialists on the interpretation of the IVDR and its complications. Clin Chem Lab Med 2021;59:491–7.
6. Medical Device Coordination Group. Guidance on classification rules for in vitro diagnostic medical devices under regulation (EU) 2017/746; 2020 [Internet]. Available from: https://ec.europa.eu/health/system/files/2022-01/md_mdcg_2020_guidance_classification_ivd-md_en.pdf.
7. International Medical Device Regulators Forum. Software as a medical device: possible framework for risk categorization and corresponding considerations; 2014 [Internet]. Available from: https://www.imdrf.org/documents/software-medical-device-possible-framework-risk-categorization-and-corresponding-considerations.
8. Arnold L, Frauch P, Klöti A, Staub M. Software assessment under consideration of validation aspects: PPS and PMS systems. Pharm Acta Helv 1998;72:327–32.
9. Medical Device Coordination Group. Guidance on clinical evaluation (MDR)/performance evaluation (IVDR) of medical device software 2020 [Internet]. Available from: https://ec.europa.eu/health/system/files/2020-09/md_mdcg_2020_1_guidance_clinic_eva_md_software_en_0.pdf.
10. Li H. Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM; 2013. Available from: http://arxiv.org/abs/1303.3997.
11. Van der Auwera GA, O'Connor BD. Genomics in the cloud: using Docker, GATK, and WDL in Terra. Sebastopol: O'Reilly Media; 2020.
12. Venkataramanan R, Swaminathan A, Prasad T, Jain A, Zuckerman S, Warty V, et al. Clinical pharmacokinetics of tacrolimus. Clin Pharmacokinet 1995;29:404–30.
13. Scholten EM, Cremers SC, Schoemaker RC, Rowshani AT, van Kan EJ, den Hartigh J, et al. AUC-guided dosing of tacrolimus prevents progressive systemic overexposure in renal transplant recipients. Kidney Int 2005;67:2440–7.
14. Ronco C, Bellomo R, Kellum JA. Acute kidney injury. Lancet 2019;394:1949–64.
15. Niemantsverdriet M, Khairoun M, El Idrissi A, Koopsen R, Hoefer I, van Solinge W, et al. Ambiguous definitions for baseline serum creatinine affect acute kidney diagnosis at the emergency department. BMC Nephrol 2021;22:371.
16. Genome in a Bottle; 2022 [Internet]. Available from: https://www.nist.gov/programs-projects/genome-bottle.
17. Wilkinson MD, Dumontier M, Aalbersberg IJJ, Appleton G, Axton M, Baak A, et al. The FAIR Guiding Principles for scientific data management and stewardship. Sci Data 2016;3:160018.

Supplementary Material: The online version of this article offers supplementary material (https://doi.org/10.1515/cclm-2022-0278).