Part I

Regulatory, Business, and Management Background
1 Introduction to Medical Software

INTRODUCTION

This chapter begins by defining what medical software is and what makes it unique, and describing the regulatory process that governs it (Section 1.1), including a brief introduction to industry standards. Following this, we discuss the constraints (both business and technical) placed on the software process by the medical environment (Section 1.2). The chapter concludes with a discussion of the challenges that result from the use of artificial intelligence/machine learning (AI/ML) techniques (Section 1.3). This section also provides references to other parts of this book where we discuss AI/ML issues.

1.1 Medical Software and the Regulatory Process

1.1.1 What is Medical Software?

Medical software is software that is used in a clinical setting, either as part of a medical device or as a standalone application. An example of the former is software that manages the delivery of radiation in a radiation treatment machine in a therapeutic radiology setting (e.g. lung radiotherapy). In this case, the software controls a device, whose misuse can cause serious harm to the patient. This example is not chosen at random! The reader may recognize that this is precisely the setting of the Therac-25 [154–156] incidents in the 1980s that precipitated the involvement of the Food and Drug Administration (FDA) in regulating medical software. We discuss the case of Therac-25 in more detail in Section 2.1.5 and the Vignette presented in Chapter 17.

At the other extreme, we may have software for analysis of patient medical images by a radiologist for diagnostic purposes. In this second scenario, the danger to the patient is potentially less serious, as the software does not interact with the patient directly. However, an error in the analysis can cause an error in the diagnosis, which may also lead to harm caused by using the wrong treatments later in the process. This is a pure software device (no hardware), formally known as SaMD – software as a medical device.

Interestingly, a recent guidance document from the Health Sciences Authority of Singapore [234] divides software that is not part of a device into three categories: (1) standalone software, (2) standalone mobile applications, and (3) web-based software. This reflects the additional challenges posed by mobile and web-based applications.
In our view, medical software is a field that lives in the intersection of five areas:
(1) medicine, (2) biomedical engineering, (3) computer science (software engineering),
(4) law, and (5) business and management. This is illustrated in Figure 1.1.

1.1.2 Some Unique Aspects of Medical Software

In the United States, most medical software (some software is exempt, as discussed in Section 2.1.1) and medical devices require clearance or approval by the FDA prior to use in patient care.\(^1\) Hence, the usual software development process of (1) design, (2) implement, (3) test, and (4) market has an additional step prior to “market,” which is to obtain regulatory clearance. As an aside: the terms “clearance” and “approval” are used for different types of regulatory processes – see Section 2.3 for more details. We will sometimes use the terms “approved”/“approval” loosely to indicate either of these processes, that is, that the product is “blessed” by the FDA to be used for patient care.

This makes medical software design and development a regulated activity that must be performed in accordance with specific laws passed by Congress and regulations issued by the FDA.\(^2\) Therefore, while in some software work the explicit writing of system specifications is considered a good idea but not mandatory,\(^3\) in the medical realm written specifications are explicitly required. Specifically, the FDA’s Quality Systems Regulations (QSR) [69] (Section C – Design Controls) states in no uncertain terms that: “The design input requirements shall be documented and shall be reviewed and approved by a designated individual(s).”

In general, designing medical software in a way that meets regulatory criteria requires more than validation of the completed software. The structure and the
environment in which the software development takes place are also critical, as discussed in Chapter 4 on quality management systems. We must be able to establish procedures and plans that demonstrate that the overall software development process is “controlled” and follows an actual plan rather than ad-hoc programming.

As a helpful analogy, consider the case of an “FDA-regulated” restaurant. The FDA is also interested in the process used to develop the food (software), not just whether it is safe (and presumably tastes good). This means that the restaurant needs to be organized in such a way that good and healthy food is likely to be made. This includes having formal processes for selecting recipes, training staff, testing for quality and how the staff goes about cooking and serving the food, and managing risk. The FDA will want to know whether the food is safe and tasty and be sure that the utensils will not cause harm to the customer (patient). Finally, the FDA can inspect a restaurant at any time to verify compliance (not just taste the food!). International standards (see Section 1.1.4) such as ISO 9001 [131] are increasingly placing more emphasis on the organization as a whole rather than narrowly focusing on the individual product.

1.1.3 The FDA and Software

The FDA’s involvement with software began with the Therac-25 incidents in the 1980s, when a software bug resulted in the delivery of excess radiation to patients by a radiotherapy treatment device, leading to severe injuries and deaths [154, 156].

The classic FDA guidance document on software is titled General Principles on Software Validation (GPSV) [70]. The GPSV describes at length the FDA’s experience with software and its suggestions (which, while not binding, one would be wise to treat as more than mere recommendations) for how software should be developed and validated. We will discuss this in more detail in Section 2.2, but one point is particularly worth highlighting in this introductory chapter. After a very interesting discussion of the differences between software and hardware, the GPSV states: “For these and other reasons, software engineering needs an even greater level of managerial scrutiny and control than does hardware engineering.” This should be eye-opening to those (especially at the managerial level) who think of software as easier or less critical than hardware. The FDA’s experience of the matter suggests the opposite.

More recently, the FDA and the International Medical Devices Regulators Forum (IMDRF) have created additional guidelines specifically for the case of standalone software under the umbrella term of SaMD [92, 125–127]. Such software is becoming increasingly common (see Chapter 7).

1.1.4 Industry Standards

While the FDA and its counterparts in other countries are the legally responsible regulatory bodies that clear/approve software for medical use, much of the specific requirements for structuring the organization, project management, software design,
implementation, verification, and validation derive from industry standards created by organizations such as the International Standards Organization (ISO) [135]. As stated on its website, the ISO is an independent nongovernmental international organization based in Geneva, Switzerland. Its membership consists of 164 national standards bodies, including ANSI – the American National Standards Institute. The ISO was founded in 1946 [135] with the participation of 25 countries. It produces standards on a variety of topics such as (1) quality management systems, (2) environmental management systems, (3) health and safety standards, (4) energy management standards, (5) food safety standards, and (6) IT security standards. A standard is defined on the ISO website as “a formula that describes the best way of doing something ... Standards are the distilled wisdom of people with expertise in their subject matter and who know the needs of the organizations they represent – people such as manufacturers, sellers, buyers, customers, trade associations, users or regulators.”

These standards represent sector-specific best practices. They are written by industry workgroups that include representatives from manufacturers and often regulators.

Relation between Regulations and Standards
While compliance with such standards is voluntary, many regulators recognize compliance with specific standards as evidence of “proper work.” For example, consider the case of the standard IEC 62304, titled “Medical device software – software life cycle processes” [2, 119]. In a document from the IMDRF [128], “Statement regarding use of IEC 62304:2006 ‘Medical device software – software life cycle processes’” [128], one can find statements from different member organizations (i.e. regulators from different countries) relating to the applicability of these standards. To quote two examples, the first from the US FDA and the second from the China Food and Drug Administration (CFDA):

IEC 62304:2006 is recognized by the US FDA medical device program as a consensus standard for which a person may submit a declaration of conformity in order to meet a premarket submission requirement or other requirements to which a standard is applicable. US FDA by recognizing IEC 62304:2006 is acknowledging that the process activities and tasks identified in this standard when used with a good quality management system and risk management system can help assure safe design and maintenance of software used in medical devices.

The IEC 62304:2006 had been translated into China industry standard: YY/T 0664-2008 equally and implement from 2009.6.1, it isn’t mandatory standard, and just is recommended standard.4

To conclude, we will quote IEC 62304 [119] on the details of medical software. The document’s introduction section states:

As a basic foundation it is assumed that MEDICAL DEVICE SOFTWARE is developed and maintained within a quality management system (see 4.1) and a RISK MANAGEMENT system (see 4.2). The RISK MANAGEMENT PROCESS is already very well addressed by the International Standard ISO 14971.

This completes the loop back to the US FDA statement above. IEC 62304 effectively says that you need to follow these best practices in your software process, but it assumes the presence of a quality management system (we discuss this in Chapter 4) and a
1.1 Medical Software and the Regulatory Process

risk management system (see Chapter 5). IEC 62304 will be our guide and anchor in Part III of this book, where we will discuss the actual process of medical software design, implementation, and testing.

In general, while adherence to these standards is voluntary, many regulators recognize specific standards as representing appropriate/best practices, so a company that adheres to specific standards is likely to have an easier path to obtaining regulatory clearance. In addition, many specific regulations are explicitly based on industry standards. As an example, the IMDRF document on quality management systems [127] makes explicit and copious references to the industry standard ISO 13485 [132]. This is not surprising, as (1) regulatory organizations often participate in the workgroups that write the standards, and (2) the real domain expertise in particular topics often lies in those organizations that have significant experience in producing such devices.

Important Standards for Medical Software

For medical software, the key standards are probably:

- ISO 9001: Quality management systems [131];
- ISO 13485: Medical devices – quality management systems [132];
- ISO 14971: Application of risk management to medical devices [133];
- IEC 62304: Medical device software – system life cycle processes [119]; and
- IEC 62366: Medical devices – application of usability engineering to medical devices (Parts 1 & 2) [122, 124].

While most of these standards come through the ISO, other organizations also issue standards. These include the Institute of Electrical and Electronic Engineers (IEEE), the International Electrotechnical Commission (IEC), and the Association for the Advancement of Medical Instrumentation (AAMI).

For those interested in Agile methodologies the AAMI technical information report (TIR) 45, titled “Guidance on the use of AGILE practices in the development of medical device software” [1], should also be consulted (see also Section 9.2.2).

In this book we will refer explicitly to (and quote) the regulatory documents more than the industry standards. This is primarily because the former are freely available and, as such, are more easily accessible to the average student. Naturally, those readers who have significant responsibility in a medical device/medical software company should read the actual standards themselves and probably attend more advanced training sessions on how to apply them.

Data Privacy and Cybersecurity

In addition to the “standard” regulatory documents, one needs to understand the implications of the increasingly strong rules on data privacy and security as a result of regulations such as HIPAA [49] in the United States and more recently GDPR [65] in the EU. Data privacy is critical for most medical software and needs to be accounted for at the beginning of the process rather than at the end. A related issue is cybersecurity. We discuss these topics in more detail in Section 3.4.
The development of a new product can be divided into research and development (R&D). From the perspective of regulated medical software development, research can be defined as unregulated exploration of the problem. Development, on the other hand, is the regulated process of product design and construction of the solution [69, 261]. In the context of medical software development, the research phase involves the development and testing of new technology such as ML algorithms and visualization techniques. Thus, the research through which we discover the requirements for the final product is outside the regulatory process; however, much like basic scientific research, it should still be documented carefully. The development phase is the process of taking this new technology, combining it with other standard tools, and creating a new software package for patient care. It starts once the requirements for the product have been finalized.

It is important for the software designer/engineer to have a sense of the underlying business environment in which their product will live or die. Given the complexity and cost of medical products, it is worth, at the start of the project, mapping out the pathway for the project from idea to product and identifying the risks involved (“failure modes” in engineering terminology) in this path and how the product may fail.

Figure 1.2, which comes from the MIT hack medicine philosophy of healthcare entrepreneurship, illustrates these risks. Here, we plot value (vertical axis) vs. development time (horizontal axis). The key lesson from this figure is that while most companies focus on starting with the technical risk, it is most important to start with de-risking the “market risk” by really testing and validating the first customer(s) that will pay for the service or product. Quite often there is an assumption that “someone” wants the technology being created, which turns out later to be untrue – this model attempts to minimize that risk first and foremost and to continue this during the technology development process. This can be done by properly assessing user needs and establishing the right specifications for the project. Once the needs have been identified, the sequence suggests that the first mode of failure is “technology risk,” which simply means that our proposed new idea (research phase) does not work out as intended. The next step is “management risk,” which refers to a failure to manage the development of the new technology into a product with a defined customer; this may involve organizational failure or financial failures – that is, failure to raise enough money to support the process. Next, we have regulatory risk – our new product fails to be cleared by the regulatory agencies despite appearing to work satisfactorily. The cause of this might be a failure in the process and is best handled by designing the product from the ground up in a manner compliant with regulatory guidelines. Finally, we can have a fully working, regulator-cleared product that fails in the market because nobody is interested in paying for it. At the risk of repeating ourselves: The regulatory agencies are interested in the whole process, not just the final output, so the process of obtaining regulatory clearance starts with the initial design of the software as opposed to just its validation.
1.2 The Medical Domain and Software

Prioritize upfront

Time
Task
Reimbursement and Consumer Drivers
Physician and Patient Adoption
Distribution

Figure 1.2
Product development and risk management. This figure shows the different aspects of risk management for a project, ranging from technology risk (will it work?) to management risk (can we pull this off?) to regulatory risk (will the FDA allow this?) to, finally, market adoption risk (will anybody buy this?). This figure comes from a presentation by Zen Chu. Z. Chu. Hacking medicine toolkit. Grand Hack presentation, April 2019. Used with permission from the author.

Failure from market risk, therefore, is very expensive after the technology has been developed and an effort made to obtain regulatory clearance. Despite this, many medical software products are created without the designers speaking to potential customers. We will discuss these topics at length in the rest of the book.

1.2.2 The Clinical Information Technology Environment

Some software is completely standalone and does not need to communicate with the outside world in any way. For example, a calculator application is completely self-contained and does not need to import or export data from other software packages/servers (other than perhaps for supporting the operating system’s global clipboard copy-and-paste operations). While this is sometimes the case for medical systems, it is far more common that any given piece of medical software plays a role in a larger process in which the inputs may come from an electronic patient record database [14] or the clinical imaging database [208] (see section 3.3 for a description of these databases.) The software may then perform some manipulation, creating some form of output, which may also be stored in an external database for later use. The interoperability standards that govern the communication between such pieces of software create “environmental constraints” that need to be properly understood as part of the software design process. Ignoring these may result
in a great piece of software that is incapable of functioning as part of a complex workflow and is, hence, ultimately useless for the end user.

1.2.3 Understanding a Clinical Problem and User Needs

Successful design in healthcare starts with the clinical pain point that is felt either by the patient/caregiver, the provider, or the healthcare delivery mechanism; only through clinical pull can projects be effectively created to properly meet user needs. The software designer must understand the bigger context in which the need arises. In general, one must first understand (1) the underlying disease/clinical situation, (2) the current methods used to address/treat the problem if any exist, and (3) the limitations of these methods and how they cause problems to users. This will allow one to establish the potential impact of any proposed solution and to assess whether the project is worth carrying out.\(^6\)

Understanding the needs of the user will involve spending a significant amount of time observing clinical procedures and asking questions to make sure the designer truly understands the context and the needs of the user.\(^7\) One needs to learn the “vocabulary” of the users and to speak some of “their language” to be able to communicate with them in a meaningful way. A key danger is that most physicians or clinical providers are busy, so time may be limited. However, failure to properly understand the users’ needs will result in a flawed product, probably produced at great expense. Sometimes many of the users’ needs are so obvious to them that they may neglect to mention these needs. The job of the software designer (and/or the business analyst in a larger company) is to make the users’ implicit needs explicit. We discuss this topic in more detail in Chapter 11.

Finally, one needs to think hard about how to validate the proposed solution. What are the performance metrics (ideally quantitative) that can be used to demonstrate to the users’ satisfaction that the software actually does what it is supposed to do? In our experience, this is a concept alien to most physicians (who can tend to think, “I will know if it works when I use it”), but such metrics are useful for proving efficacy in the regulatory process (and for marketing, etc.).

1.2.4 Software Engineering and Medical Software

What does the process of successfully designing medical software that will meet regulatory standards actually look like?

The first and critical step, worth reiterating here, is that the process must happen within an organization that has an appropriate quality management system (QMS) [127, 131, 132, 134] for this purpose. This system forms the critical environment/substrate in which the software will be designed, developed, and produced. In the absence of this environment, one cannot (at least according to the regulatory and industry consensus – an example is the comment to this effect in IEC 62304 [119],