Congenius

Congenius Whitepaper

Risk management for software

July 2023
By Dr Dirk Hüber

Edited by Hannah Matthews

Contents

1. Introduction

Introduction

Pre-reading note: We advise that before reading this whitepaper, you review two of our previous articles regarding risk management:

Risk Analysis v FMEA: How knowing the difference benefits your risk management

How to connect failure modes and risks in a risk management file

The content of the articles above has also been explained briefly **in this video**.

In these two articles, we explain some fundamental principles of state-of-the-art risk management according to ISO 14971 which are applied in this whitepaper.

The principles outlined in the articles mentioned above also apply if the subject of the risk management activities is software, be it **Software in a Medical Device (SiMD)** – i.e., where the software is a component of a medical device, e.g., firmware or **Software as a Medical Device (SaMD)** – i.e., where the software itself is the medical device.

However, there are certain challenges and special aspects that need to be considered when undertaking risk management for software.

In the following pages, we will discuss the implications for the risk analysis, as well as for the different FMEA types. We will start with **Software FMEAs**, continue with an explanation of the consequences for **Risk Analysis**, before moving onto **other FMEA types**. We have chosen this sequence because the core focus in software risk management is on risks caused by software failures.

Finally, we will also provide some advice on **security assessments**, **Software of Unknown Provenance (SOUP)**, and **software updates based on** general requirements stated in **IEC 62304** (Software Lifecycle Management).

2. Software FMEAs

Software FMEAs

For software, the Design-FMEA is replaced by a Software-FMEA. Whilst the structure remains the same, the difference is that in a Software-FMEA software failures must be identified and evaluated, i.e., the failure modes are software failures. This might sound like only a minor, or even superficial difference, but in practice, it is often difficult for FMEA teams to understand what a software failure is, and what it is not.

The other difficulty often faced is the estimation of software failure occurrence. Indeed, often the occurrence of a software failure is simply not known.

This chapter looks at how you would handle such a situation, and still achieve meaningful risk management.

Software FMEAs

Software Failures

Let us start by explaining what a software failure is NOT:

A software failure is neither a software defect (bug) nor a software fault.

By now we have used three different terms that require definition:

Software defect	Software fault	Software failure
This is an error in the design or implementation of the software. Also known as a “bug”.	This is a software condition that causes the software not to function as intended.	This is a software condition that causes the medical device not to perform as specified.

Note the difference in the definitions for a software fault and a software failure: **it is the object that does not function as specified**. In cases where the software itself is the medical device (SaMD), “software failure” and “software fault” are equivalent. In any other cases, they are not.

As such, the failure cause may be a software fault or software defect, or a condition outside of the software:

If the cause is a software fault or a software defect, there may be a direct or an indirect cause:

- **Direct causes** are local to the software item, and do not necessarily affect other software items, for example, incorrectly implemented algorithms, errors in input processing, and errors in output processing of the software item.
- **Indirect causes** are more unpredictable than direct causes. Examples of indirect causes of software failures include stack overflow, uninitialized pointers, trace conditions, bit flips, low-power condition, or software sources such as operating systems, libraries, and Software of Unknown Provenance (SOUP).

Causes outside of the software may include user actions or unexpected behaviour of connected hardware, e.g., sensors.

Software FMEAs

Occurrence of Software Failures

The probability of a software failure occurring is often not known and cannot be estimated, even qualitatively - especially for new software. Let's look at how this situation can be handled in a Software-FMEA in a way that ensures the FMEA remains useful.

First, if the occurrence of a software failure is unknown, it must be set to the highest value defined in the risk policy (refer to definitions in IEC 62304). The detectability of the software failure, i.e., the probability of detecting that failure before the effect on the functionality of the software sets in (thus the effect might become prevented), can nevertheless always be evaluated. As such, mitigations are prioritised only against the severity of the effect and the detectability.

Since assigning the highest value defined in the risk policy for unknown probability of occurrence overestimates the resulting failure evaluation, the respective failure acceptance matrix should be adjusted accordingly. The simplest way to do this is to drop the occurrence from failure acceptance and base failure acceptance only on detectability and severity.

If the occurrence of software failures is unknown, this also influences the evaluation of the probability of a risk caused by a software failure occurring to a patient, user, or environment, i.e., the evaluation of the occurrence of the harm. We will discuss this aspect in the following pages.

It's worth noting that for **software already in the market** it is expected that occurrence of software failures is determined from post market data, for example from complaints.

Software FMEAs

Single Faults

MDR requires software to be single fault safe, i.e., no single event must lead to a software failure.

In other words, a software failure may only be caused by a chain of events or multiple events. As such, whether a software failure is caused by a single fault or not must be represented in the Software-FMEA.

The easiest way to do this is to introduce an extra column in your Software-FMEA table, where failure causes for software failures before mitigation are marked with “Single Fault [yes/no]”.

Mitigation of software failures and achievement of single fault safety requires defendable strategies in the development and architecture of the software. Here are some examples:

- Redundancy (does not mitigate systematic errors)
- Diversity (on component level)
- Self-test (does not mitigate systematic errors)
- High integrity components
- High level of rigor in the software development process
- Inherently safe design and manufacture
- Protective measures in the medical device itself or in the manufacturing process
- Information for safety, and user training

Some of these mitigations are generic in nature, i.e., they do not mitigate a specific single fault software failure, but they improve the development and manufacturing process such to reduce the possibility of single faults to occur. Such mitigations are part of the respective SOPs, so as a result, your SOPs need to address single fault safety and how it shall be achieved in the development and manufacturing process.

Independent of this, if a single fault condition is detected in the Software-FMEA, it must be mitigated with a specific measure. In order to document if a mitigation has removed the single fault condition, another column with “Single Fault [yes/no]” after mitigation is required.

3. Risk analysis

Risk analysis

If a software failure may cause a risk to a patient, user, or an environment, this risk must be analysed in your Risk Analysis. Part of the Risk Analysis is to evaluate the probability of the harm occurring. This becomes challenging if the occurrence of the software failure is not known. This chapter looks at how this situation can be handled.

As explained in our previous article How to connect failure modes and risks in a risk management file, it is sometimes useful to split the probability of the harm to occur into two separate probabilities:

1. **the probability of the hazardous situation to occur (p1), and**
2. **the probability of the harm to occur if the hazardous situation has occurred (p2)**

In this case, the probability of the harm to occur if the hazardous situation has occurred can always be determined. However, if the probability of the software failure to occur is unknown, then the probability of the hazardous situation to occur is also not known. As such, it must be assigned the highest value as defined in the risk policy (refer to definitions in IEC 62304). Accordingly, risk mitigations are only prioritised by the severity of the harm and the probability of the harm to occur if the hazardous situation has occurred.

As with software failures, the corresponding risks are overestimated, and your respective risk acceptance matrix should be adjusted for this. In this case, the simplest way to do this would be to draw the risk acceptance matrix for risks caused by software failures only against the probability of the harm to occur if the hazardous situation has occurred (p2), and thus reduce the probability of the hazardous situation to occur (p1).

Software already in the market

It’s important to note again, that for software in the market it is expected that the probability of the harm to occur is determined from post market data, for example from complaints.

4. Usability FMEAs & Process FMEAs

Usability FMEAs & Process FMEAs

Usability-FMEA

If a software has a user interface, a Usability-FMEA must be conducted.

All other aspects of usability also apply. Bear in mind, that a user interface might be indirect, i.e., not via a screen and/or physical or virtual keys or keyboard. An example for an indirect user interface would be a report that the software generates, addressed to a certain user group.

Process-FMEA

A software has a manufacturing process that comprises the build & release process. As such, a software also requires a Process-FMEA that analyses the build & release.

The distribution of the software is crucial, especially in case of updates. This distribution process should also be analysed in an FMEA and might be included within your Process-FMEA. However, distribution, especially when done remotely rather than solely in a safe and protected environment (e.g., off-line), must also be part of the cybersecurity assessment.

5. Cybersecurity assessment

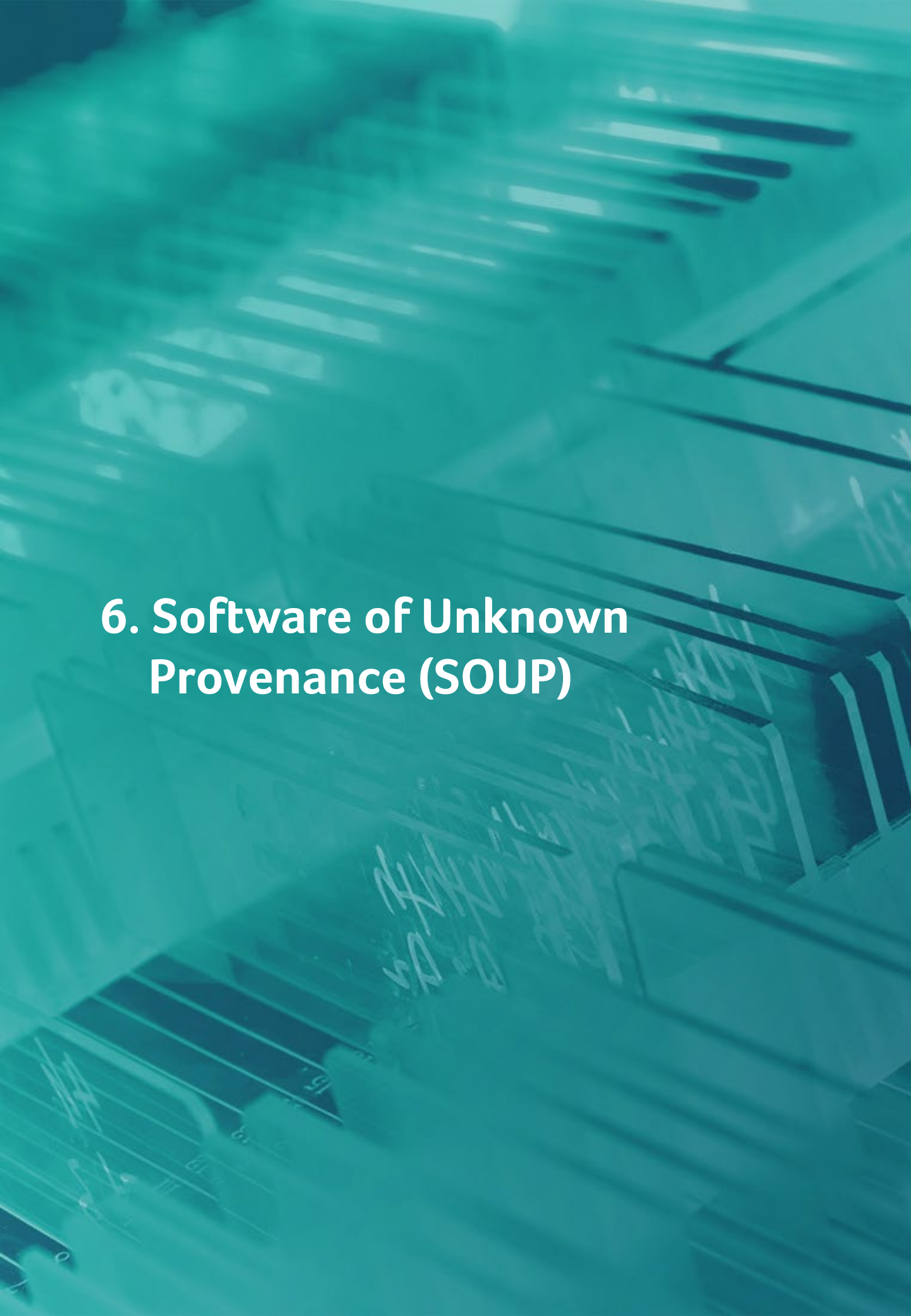
Cybersecurity assessment

We will touch here only briefly on the cybersecurity assessment according to IEC 62304, since this is a separate and special topic.

- The scope of a security assessment is to identify risks (or threats) to data integrity and data safety, but not patient or user risks. However, a risk to data integrity or data safety may cause a risk to the patient or user. Hence, such risks must be escalated to the risk analysis, where the risk to patient or user is evaluated and mitigated (analogously to design or use failures – see our aforementioned articles for more on this).

- Note that physical security must also be assessed, e.g., physical access to the server location or other critical hardware.

For more information on cybersecurity for medical devices, read our whitepaper here.



6. Software of Unknown Provenance (SOUP)

Software of Unknown Provenance (SOUP)

All medical device software contains some SOUP, as it simply doesn't make sense to develop every aspect from scratch. Software developers will inevitably use what is already available to create efficiencies. That said, the Legal Manufacturer has the full responsibility for any SOUP used within their medical device.

Whilst SOUP usually lacks adequate design documentation, it still requires the performance of a Software-FMEA and a Security Assessment.

For any SOUP that is part of a medical device software, its origin and the circumstances surrounding the software documentation must be explained.

For your Software-FMEA

The functional and performance requirements of the SOUP, and hardware and software that is necessary for the proper function of the SOUP must be identified and then analysed in the Software-FMEA. Additionally, any risks resulting from the incomplete documentation must be analysed. If available, lists from the manufacturer of the SOUP containing known issues shall be used as well in order to identify potential software failures.

For your Security Assessment

When it comes to your Security Assessment, it's essential to identify how the SOUP is integrated into the software system (interfaces to other software items), as well as the architecture of the software system. Again, any security related information and maintenance policy of the SOUP available from the manufacturer of the SOUP must be considered.

Important note

SOUP must be tested against the functional and performance requirements, and its proper integration into the whole software system. SOUP must also be part of your broader security testing.

7. Software updates

Software updates

For software updates, you need to distinguish between updates that fix bugs and mitigate identified risks (especially for urgent, safety relevant updates) but do not change the functionality of the software, and regular updates, that improve or enhance the functionality of the software.

Whereas regular updates will follow the standard development process with respective test and verification activities, safety relevant or otherwise urgent updates require a fast-track process that nevertheless ensures that the updated software is safe and functions according to its specifications. We recommend having a process in place within your QMS for such a fast-track update.

Regardless of the reason for a software update, it always results in a change to the software, and a change process must therefore be followed (including in the case of a fast-track update). For software of classes B or C according to IEC 62304, the change process must include a check for whether the changes planned for the software interfere with any existing risk mitigations in the risk management file. If so, appropriate actions must be taken that ensure the mitigations are still effective. The affected risk mitigations in the risk management file must then be updated accordingly.

The check for whether the changes planned for the software interfere with any existing risk mitigations in the risk management file is done most efficiently if all software specifications that are based on a mitigation of a risk in the risk analysis, a failure in an FMEA, or a security risk in the security assessment, are traced to the respective risk or failure. If such a specification changes, interference occurs. If the specifications remain consistent, interference is avoided. However, in any case, testing of the changed software should be extended thus far to make the detection of such an interference likely in case it was overseen in the check.

Independent of the above requirement for changes to class B or C software, for any change you must check whether your risk management file needs updating, especially if the changes may cause new failures or risks. This requirement is the same as that for changes to the hardware of a medical device.

Congenius

For more information on Risk management for software, feel free to get in touch with our Quality team.

Congenius AG
Riedstrasse 1
CH - 8953 Dietikon

E: info@congenius.ch
T: +41 44 741 04 04

congenius.ch

Revision 1 – July 2023

© 2023 Copyright Congenius AG. All Rights Reserved