iss
INTEGRATED SCIENTIFIC SERVICES
A MEDTECH COMPANY

IEC 62304 Software Development

A Real-World Perspective

Sharpen your Skills 2020, 28.04.2020

ISS Integrated Scientific Services AG

Matthias Steck, Senior Software Engineer

matthias.steck@iss-ag.ch

iss
INTEGRATED SCIENTIFIC SERVICES
A MEDTECH COMPANY

IEC 62304 Versions

IEC 62304:2006 / EN 62304:2006	Edition 1.0	Harmonized for MDD
IEC 62304:2006/Amd1:2015 / EN 62304:2006/Amd1:2015	Edition 1.1	FDA Recognized Consensus Standard State of the art
IEC 62304:20xx	Edition 2.0	Work in progress

Basic Concepts

Software is anything that is executed on a processor or by an interpreter; its the thing you cannot touch but which makes the other thing that you can touch do the thing you want*:

- Application Software on a PC
- Mobile App on a phone
- Firmware / Embedded Software inside a device
- FPGA Source Code – Configurable hardware that is "programmed" (as of Amd1:2015)

* the computer does what you tell it to do, not what you want it to do

Embedded software or **Medical Device Software** is software that is part of a physical medical device and is designed for a specific hardware.

Standalone software or **Software as a Medical Device (SaMD)** is software that is not part of a physical medical device and is designed to run on a generic hardware/software platform. The software is a medical device in itself.

Where the platform is the combination of hardware and software that is not part of the health software, but is required to run it; e.g. operating system, system libraries.

Note: The Medical Device Coordination Group (MDCG) uses the term Medical Device Software (MDSW) for both, embedded and standalone software.

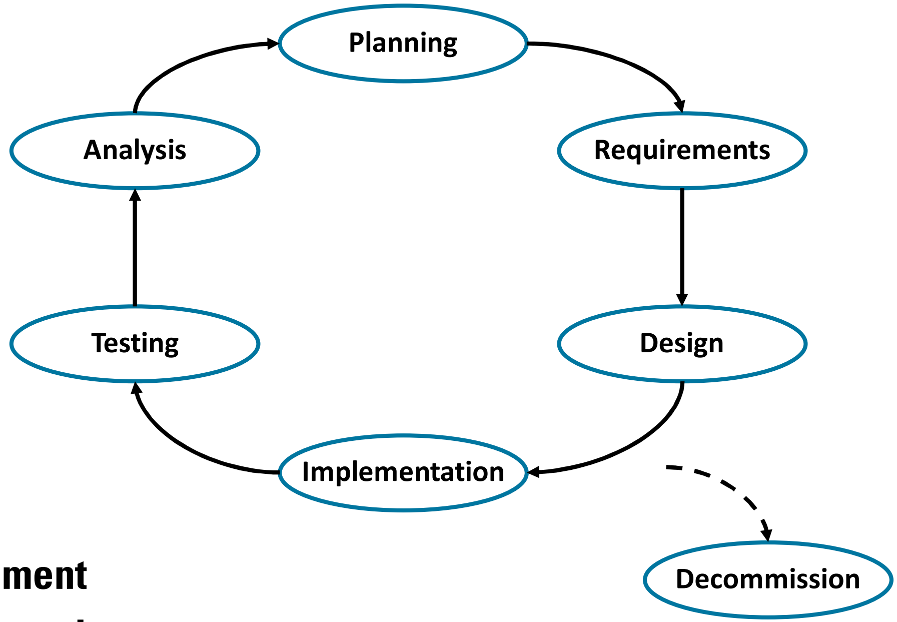
SOUP (Software of Unknown Provenance)

- Software item that is already developed and generally available and that has not been developed for the purpose of being incorporated into the medical device (also known as “off-the-shelf-software” or “3rd party component”).
- Software item previously developed for which adequate records of the development processes are not available

Note: Usually integrated into a product to reduce development time and cost and/or not to re-invent the wheel.

Note: The standard defines requirements and activities regarding the use of SOUP, adding SOUP thus also incurs some effort and costs.

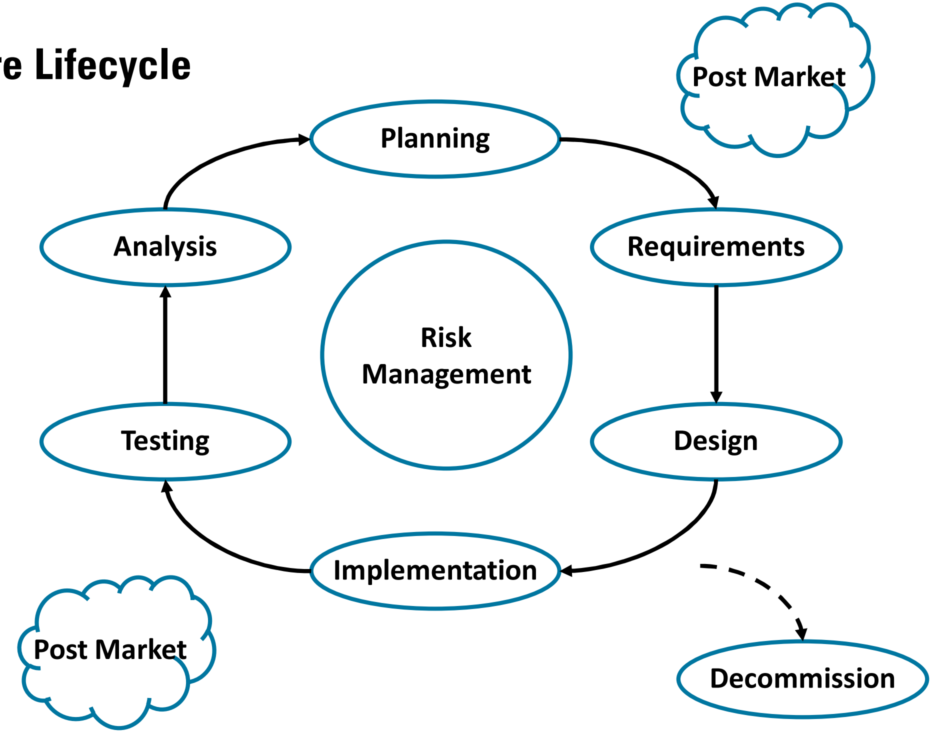
Product Lifecycle



First loop is **development**

Subsequent loops are **maintenance**

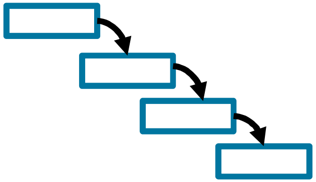
Medical Software Lifecycle



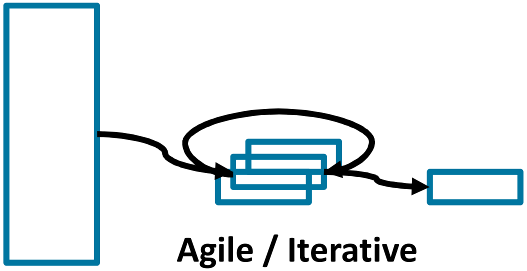
Development Models

- Multiple ways to develop software exist, each one with advantages and disadvantages
- Not all are suited for all environments; e.g. medical devices
- People can get a bit religious about them

V-Model



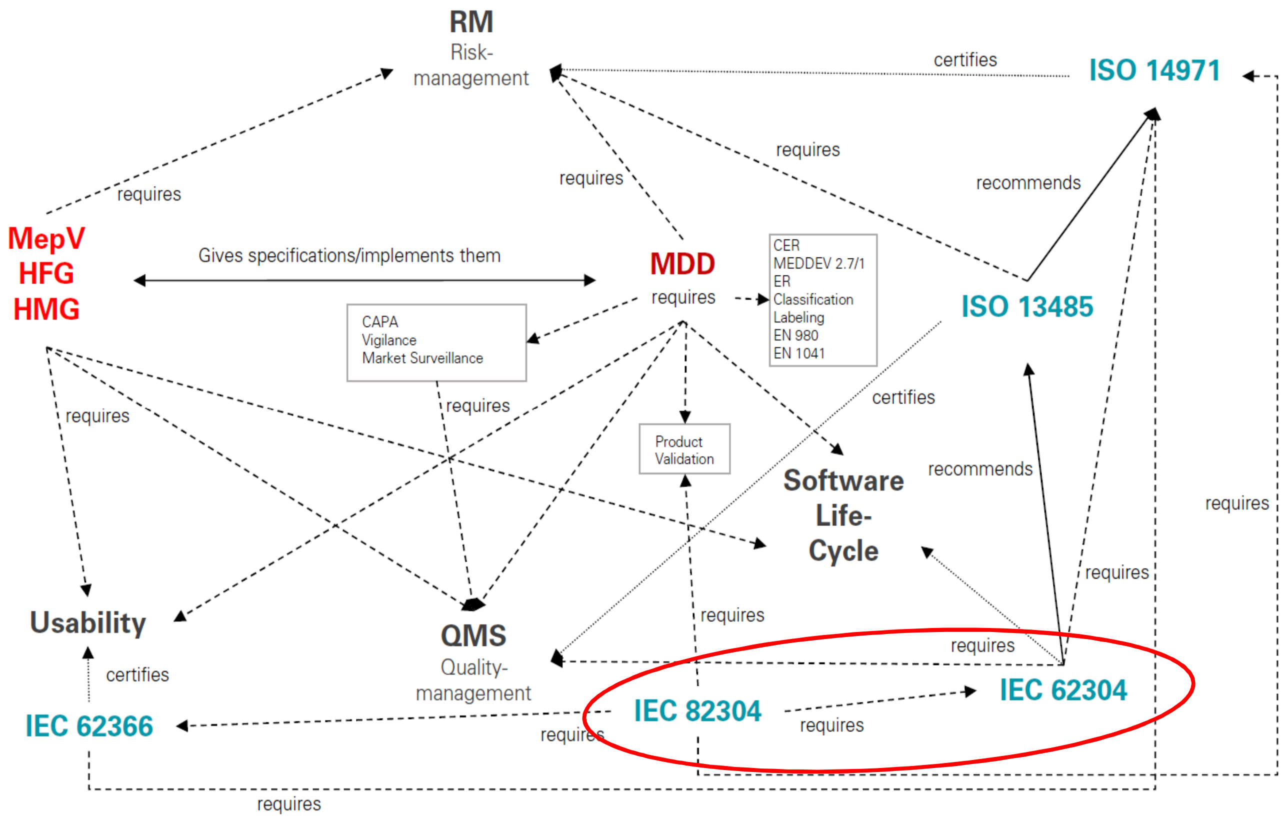
Waterfall



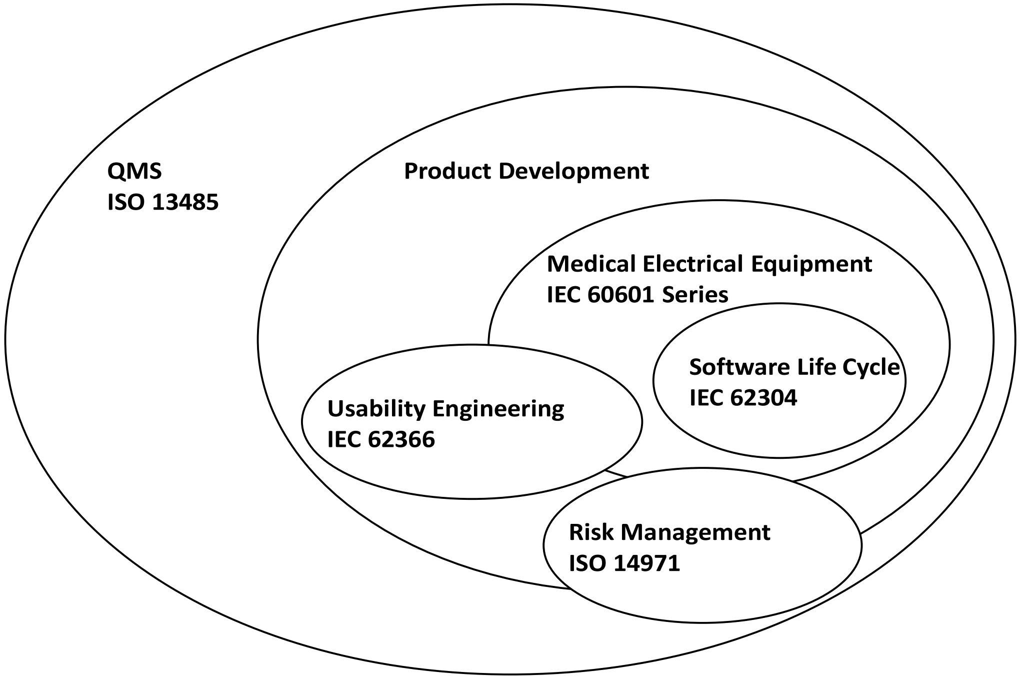
Agile / Iterative

IEC 62304 / EN 62304 at a Glance

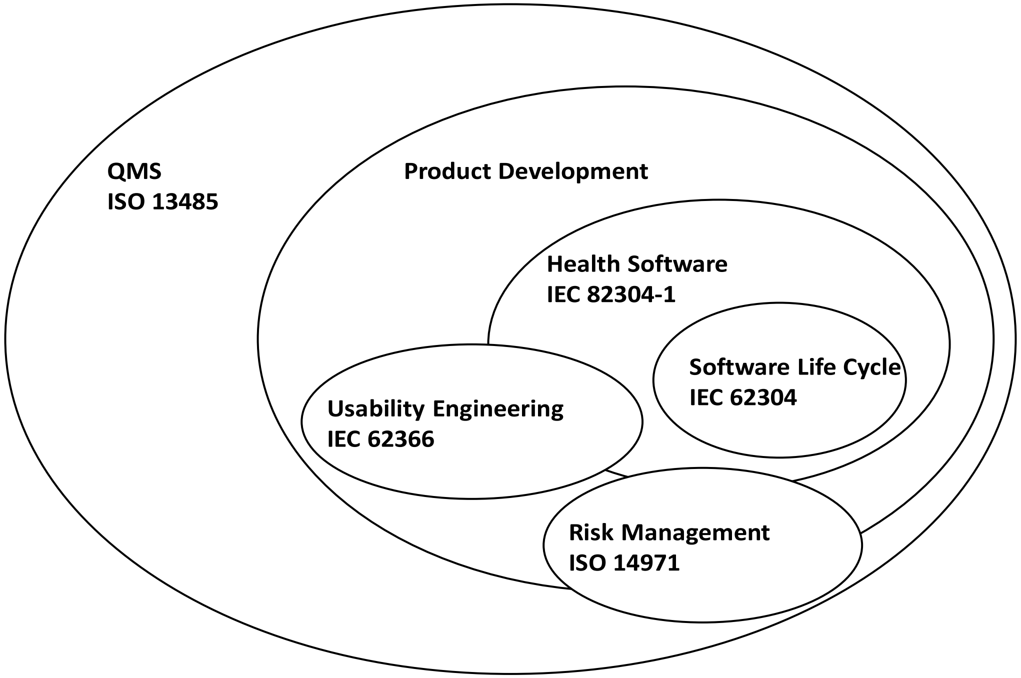
- The IEC 62304 is a process standard, it defines requirements to the development but not the product itself.
- Evidence of the correct application of the standard, i.e. performing the required activities, is the documentation
- Does not want to force a development model / process (e.g. Waterfall, V-model, SCRUM), but…


RM
Risk-management
certifies
ISO 14971
requires
requires
recommends
requires
MepV
HFG
HMG
Gives specifications/implements them
MDD
requires
CER
MEDDEV 2.7/1
ER
Classification
Labeling
EN 980
EN 1041
ISO 13485
CAPA
Vigilance
Market Surveillance
requires
requires
certifies
Product Validation
Software Life-Cycle
recommends
requires
requires
Usability
QMS
Quality-management
requires
requires
certifies
IEC 62366
IEC 82304
IEC 62304
requires
requires
requires

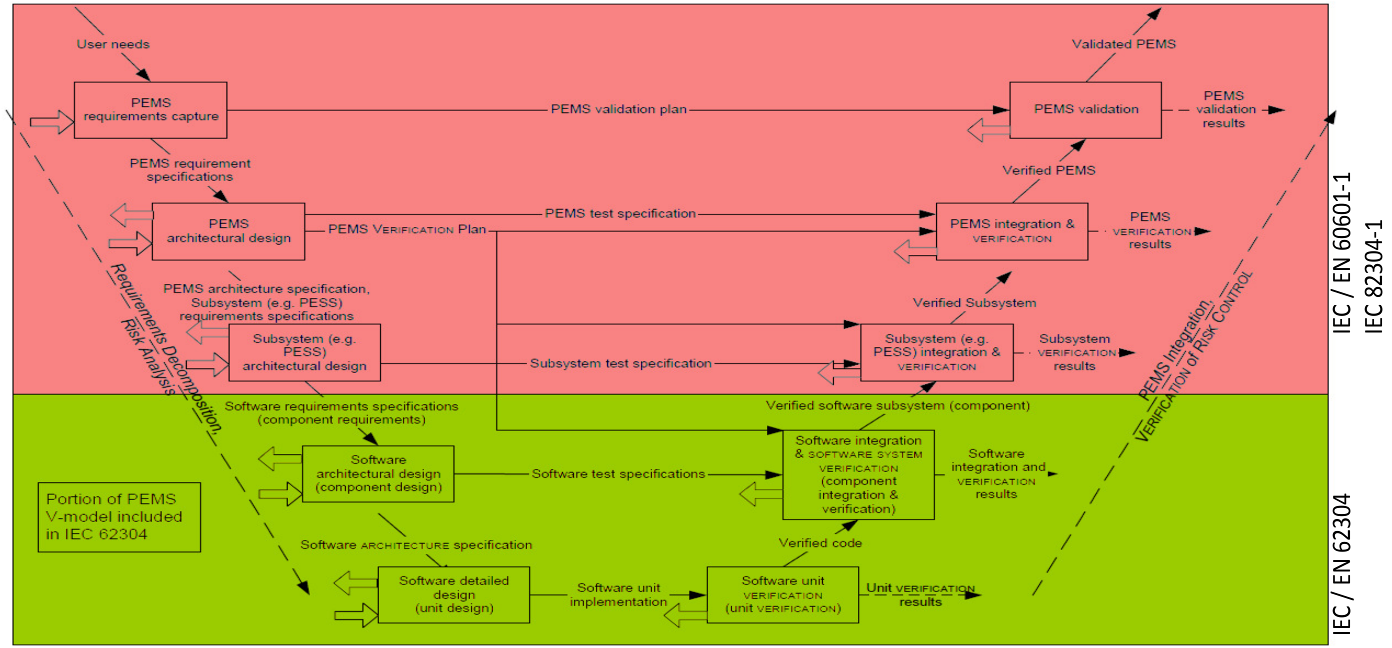
Different View (embedded software)



Different View (standalone software)



Coverage of IEC 62304 and IEC 82304-1



User needs
PEMS requirements capture
PEMS validation plan
PEMS validation
Validated PEMS
PEMS validation results
PEMS requirement specifications
PEMS architectural design
PEMS test specification
PEMS VERIFICATION Plan
PEMS integration & VERIFICATION
Verified PEMS
PEMS VERIFICATION results
PEMS architecture specification, Subsystem (e.g. PESS) requirements specifications
Subsystem (e.g. PESS) architectural design
Subsystem test specification
Subsystem (e.g. PESS) integration & VERIFICATION
Verified Subsystem
Subsystem VERIFICATION results
Requirements Decomposition, Risk Analysis
PEMS Integration, VERIFICATION of RISK CONTROL
Software requirements specifications (component requirements)
Software architectural design (component design)
Software test specifications
Software integration & SOFTWARE SYSTEM VERIFICATION (component integration & verification)
Verified software subsystem (component)
Software integration and VERIFICATION results
Portion of PEMS V-model included in IEC 62304
Software ARCHITECTURE specification
Software detailed design (unit design)
Software unit implementation
Software unit VERIFICATION (unit VERIFICATION)
Verified code
Unit VERIFICATION results
IEC / EN 60601-1
IEC 82304-1
IEC / EN 62304

Risk Based Approach

The standard defines three software safety classes:

- **Class A**: No injury or damage to health is possible
- **Class B**: Non-serious injury is possible
- **Class C**: Death or serious injury is possible

The software safety class applies to the software as a whole, but may optionally be applied to individual or groups of software modules or components, e.g. low-risk components may fall into a lower safety class.

The software safety class is defined by the potential harm emanating from the software, regardless of the probability of the harm actually occurring.

Note: Software safety class is independent from the medical device classification (Europe) and the level of concern (FDA); all of them are risk-based though.

The software safety class determines which lifecycle activities need to be performed for the specific part of the software:

IEC 62304:2006/Amd.1:2015				
Section	**Title**	**Class A**	**Class B**	**Class C**
5	***Software development PROCESS***			
5.1	**Software development planning**			
5.1.1	Software development plan	X	X	X
5.1.2	Keep software development plan updated	X	X	X
5.1.3	Software development plan reference to SYSTEM design and development	X	X	X
5.1.4	Software development standards, methods and tools planning			X
5.1.5	Software integration and integration testing planning		X	X
5.1.6	Software VERIFICATION planning	X	X	X
5.1.7	Software RISK MANAGEMENT planning	X	X	X

The software safety class should be determined through risk and hazard analysis:

Residual risks only related to device software						
		Severity				
		Class A	Class B			Class C
		A	B	C	D	E
Probability	5					
	4					
	3	5				
	2	1				
	1	4		8		

INTEGRATED SCIENTIFIC SERVICES
A MEDTECH COMPANY

Software Safety Classification

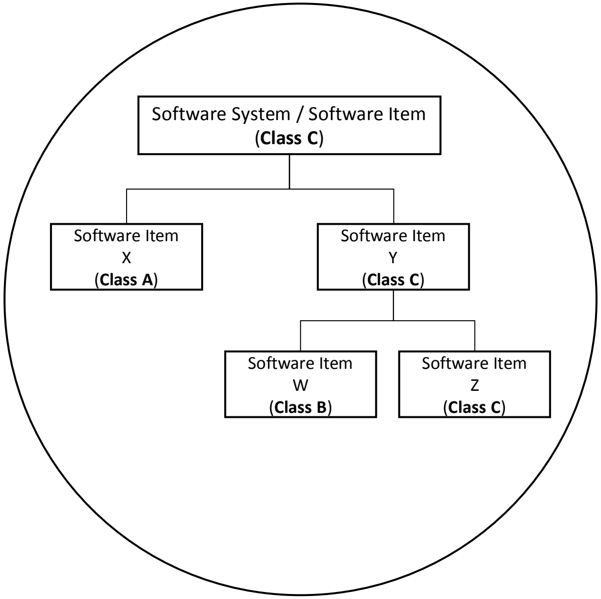
- As early as possible as it determines the activities required
- If not known (e.g. risk management not done yet) assume safety class C
- For the residual risk, only consider risk control measures external to the software
  The standard essentially assumes that if the software fails, it fails catastrophically, i.e. negating software-based risk control measures.

Note: The standard states that "Probability of a software failure shall be assumed to be 1", this only concerns the software safety classification and not failure probability for risk management or in general

Software Safety Classification

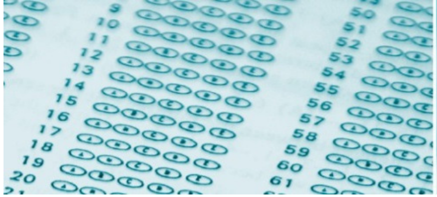
Software items with a lower safety class must not be able to adversely affect software items of a higher safety class; the manufacturer must provide rationale for the segregation between such items. If this is not possible, the items cannot be in a lower safety class.

Also software-based risk control measures take the safety class of the risk they're controlling.



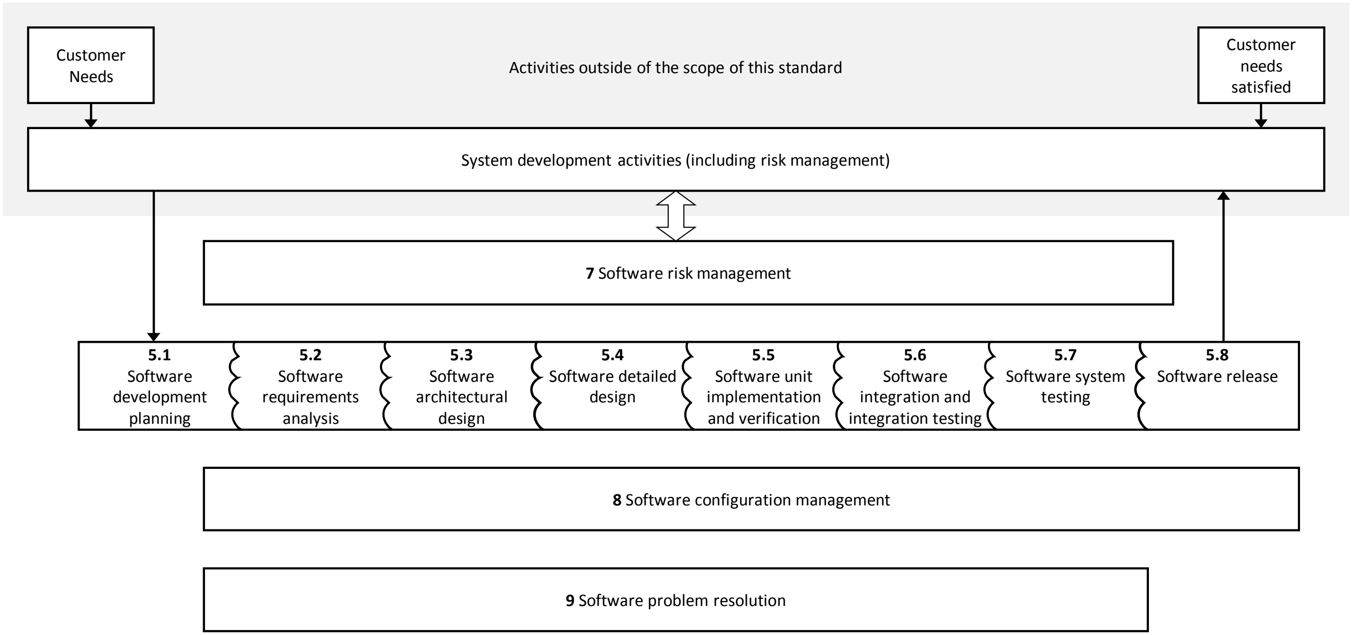
Processes defined by the IEC 62304:

- Software development (section 5)
- Software maintenance (section 6)
- Software risk management (section 7)
- Software configuration management (section 8)
- Software problem resolution (section 9)

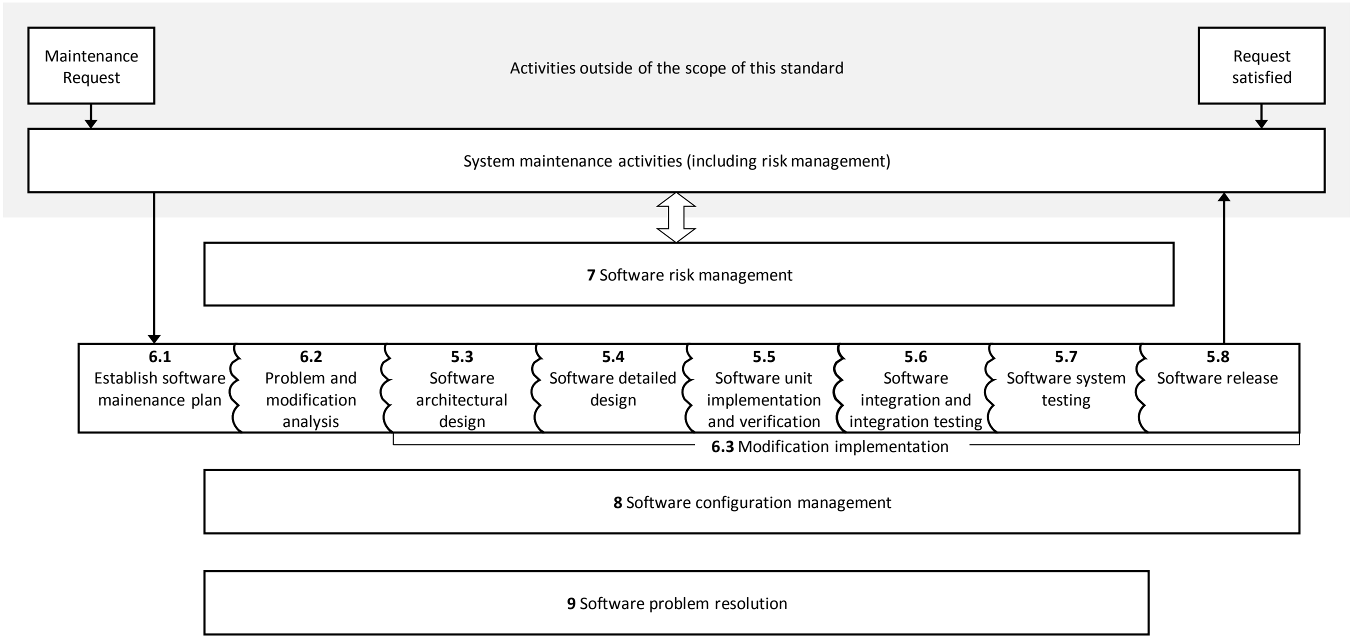


$$\begin{tikzcd}\textbf{Plan} \arrow[r] & \textbf{Do} \arrow[r] & \textbf{Verify}\end{tikzcd}$$

Software Development Process


Customer Needs
Activities outside of the scope of this standard
Customer needs satisfied
System development activities (including risk management)
7 Software risk management
5.1 Software development planning
5.2 Software requirements analysis
5.3 Software architectural design
5.4 Software detailed design
5.5 Software unit implementation and verification
5.6 Software integration and integration testing
5.7 Software system testing
5.8 Software release
8 Software configuration management
9 Software problem resolution

Software Maintenance Process



Maintenance Request
Activities outside of the scope of this standard
Request satisfied
System maintenance activities (including risk management)
7 Software risk management
6.1 Establish software mainenance plan
6.2 Problem and modification analysis
5.3 Software architectural design
5.4 Software detailed design
5.5 Software unit implementation and verification
5.6 Software integration and integration testing
5.7 Software system testing
5.8 Software release
6.3 Modification implementation
8 Software configuration management
9 Software problem resolution

Software Risk Management Process

Risk Management with a software perspective:

- Identify software items that could contribute to a hazardous situation and identify the potential causes
- Define and implement risk control measures
- Verify risk control measures (implemented and effective)
- Performed during development and maintenance activities; e.g. design changes must be analyzed for their impact on the patient risk (new risks, impact on existing risks or risk control measures)

Note: IEC 62304 mandates risk management according to ISO 14971

Note: Software developers are not risk managers! They can and should be part of the risk management team as subject matter experts though.

Configuration Management Process

Identify the inputs (and environment) from which a specific output has been developed:

- Establish means to identify configuration items (traceability)
- Techfile documents as defined in the document plan
- Source code as maintained in the version control system
- Toolchain, SOUP, and other inputs as defined in the configuration manual

Note: The configuration of a software release is essentially a snapshot of all the inputs and the generated output, including all relevant documents of the techfile.

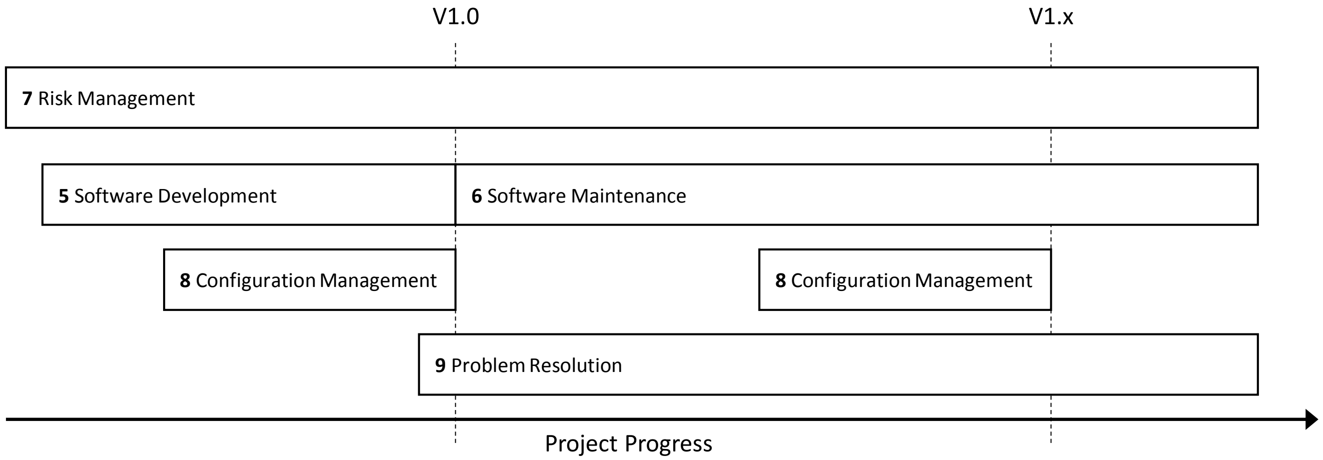
Problem Resolution Process

Handle problems found during development or in the field:

- Prepare problem report for each problem found
- Investigate the problem and analyze the problem's impact on safety (risk management, not the developer)
- Inform relevant parties
- Create change requests (change management, maintenance process) to fix the problem
- Verify problem resolution (also add to regression tests)
- Analyse problems for trends

Note: Most of these activities can be handled by a properly configured ticketing system.

Chronological order often encountered in projects



- Configuration management is neglected during development and maintenance and rushed before a release (document signing marathon)
- Problem resolution processes usually start when the software release is almost done and fails some minor tests; e.g. defects are entered into the ticketing system and fixed in a future release

IEC 62304 in the Field, as seen by a Software Development Consultant

Types of Projects

- Green Field vs Brown Field
- Motivated vs non-Motivated
- Fast Exit vs becoming Legal Manufacturer
- Development processes applied vs “uh, something like scrum?”

Note: Usually people only call for external help when they’re already stuck

ISS
INTEGRATED SCIENTIFIC SERVICES
A MEDTECH COMPANY

Example – “University Project”

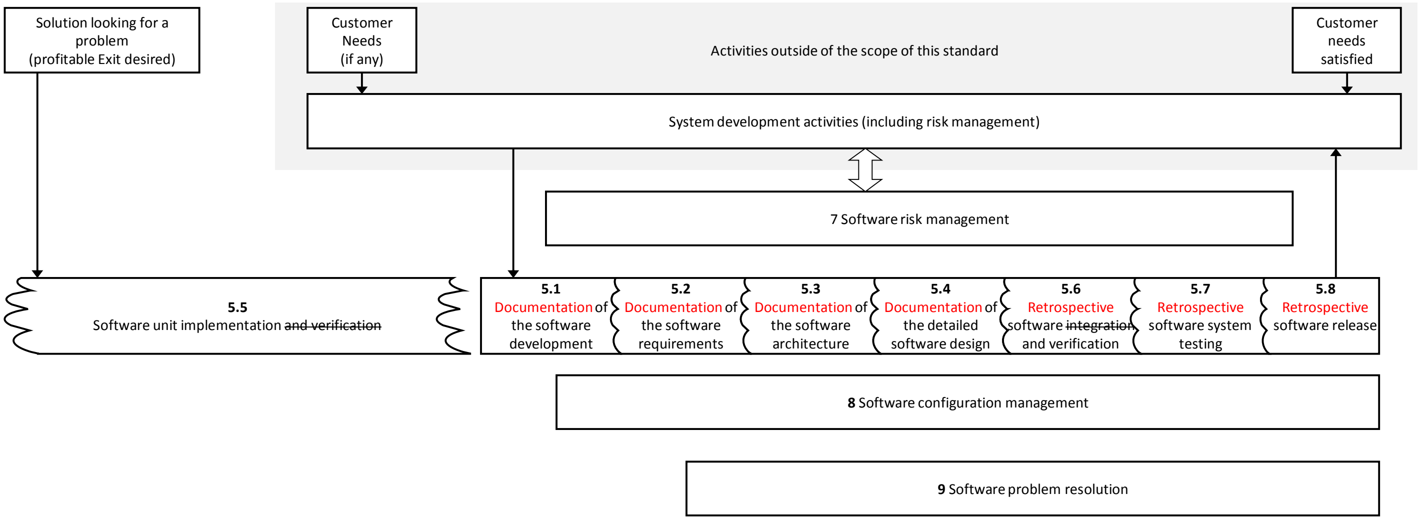
- Develops a proof of concept and wants to market it
- High employee turnover (e.g. built by postdocs and research assistants)
- Does not employ a structured development process
- No experience in medical device development / medical device software development
- No reasonable way to make the existing result compliant with the regulatory or normative requirements
- Project abort or redo from start by an experienced supplier or manufacturer

Note: To make things worse, proof of concept sometimes already used on patients through university clinics or similar institution.

Example – "Startup, not motivated"

- Develops a product and wants to market it
- Suddenly confronted by fact that product is a medical device and covered by legislative and normative requirements (and that these also apply to startups)
- Goal is quick exit and not becoming a legal manufacturer
- Retrospective application of IEC 82304/IEC 62304 activities and documentation; find gaps and fill them
- IEC 82304/IEC 62304 outsourced / ghostwriting by service provider
- No or negligible internal acquisition of know-how
- Painful but possible for Class I products (MDD, likely no longer possible under MDR)

Retrospective documentation and execution of lifecycle activities


Solution looking for a problem (profitable Exit desired)
Customer Needs (if any)
Activities outside of the scope of this standard
Customer needs satisfied
System development activities (including risk management)
7 Software risk management
5.5 Software unit implementation ~~and verification~~
5.1 Documentation of the software development
5.2 Documentation of the software requirements
5.3 Documentation of the software architecture
5.4 Documentation of the detailed software design
5.6 Retrospective software ~~integration~~ and verification
5.7 Retrospective software system testing
5.8 Retrospective software release
8 Software configuration management
9 Software problem resolution

Example – “Startup, motivated”

- Develops a product and wants to market it
- Suddenly confronted by fact that product is a medical device and covered by legislative and normative requirements
- Goal is a medium- or long-term exit (e.g. after clinical trial) or becoming a legal manufacturer
- Retrospective application of IEC 82304/IEC 62304 activities and documentation; find gaps and fill them
- Introduction of quality management system
- Requires external help, often also to acquire the required knowhow
- May require changes to the product
- Painful but possible

INTEGRATED SCIENTIFIC SERVICES
A MEDTECH COMPANY

Example – “Legacy”

- Existing product, possibly on the market for a long time
- New features or regulatory changes makes it a medical device
- Retrospective application of IEC 82304/IEC 62304 activities and documentation; find gaps and fill them; reverse engineering may be required (legacy process)
- Introduction of quality management system or transformation / extension of existing QMS (e.g. ISO 9001 to ISO 13485)
- Usually requires external help, often to acquire the required knowhow
- Painful but possible

Note: The legacy software process can only be applied for software that has already been placed on the market legally.

What does work

- Compliant development from the start
- Retrospective activities and documentation for legacy projects or motivated startups
- Remove medical functionality and market product as lifestyle product or medical device of a lower class
- Using the time the product is sold as a lifestyle product to get the required processes and infrastructure to create medical devices, collecting and using post market data from the lifestyle product and in the end add the medical device functionality back to the product and release it as such
- Moving from pure agile development to IEC compliant development

ISS
INTEGRATED SCIENTIFIC SERVICES
A MEDTECH COMPANY

What usually doesn't work

- Shadow-Techfile / Ghostwriting
- University → CE Mark / University as a legal manufacturer
- Pure agile software development (e.g. pure SCRUM)
- Forcing compliant software development processes on developers that don't want to use them and/or don't understand the reasons (e.g. when moving from SCRUM to IEC 62304 development processes)
- Replacing consultants until you find one that tells you that you don't have to do change anything…

Medical Device Software Development – The Right Mindset

- The goal is to achieve the intended use with minimal risk to the patient
- Processes are here for a reason and not just to torture the developer
- Software development is one part of a whole; no lone guns; changes to the software design may have impact on e.g. risk management, usability engineering, etc…
- Design changes are not local; no tunnel vision
- Error handling vs Essential Performance
- Design changes only through the software maintenance process
- SOUP is not free (there ain’t no such thing as a free lunch)
- SOUP – don’t forget software licenses (e.g. open source)
- Technical debt is even worse than for “normal software”

IEC 62304 and Agile Development

While the IEC 62304 does not want to promote a specific development model, it strongly leans towards the V-model. Today software is often developed iteratively (agile software development); which can be done compliant to the IEC 62304:

- Interpret the V-model of the standard as **document landscape** and not as fixed development process
- Create and release **software development plan** and **document plan** at the start of the development
  - Update draft versions continuously
  - Release as new version on major changes
- Continuously update all documents, **software requirements and design must be released before testing** (Verification)
- Plan, perform and document **reviews**
- Integrate **risk management** into processes

IEC 62304 and Agile Development

- The scope of the IEC 62304 is wider than that of most agile methodologies. The agile process need to be expanded with the additional activities
- The AAMI Technical Report **TIR 45** provides further information on how to implement IEC 62304 compliant agile development
- Human factor; don’t leave the developers behind

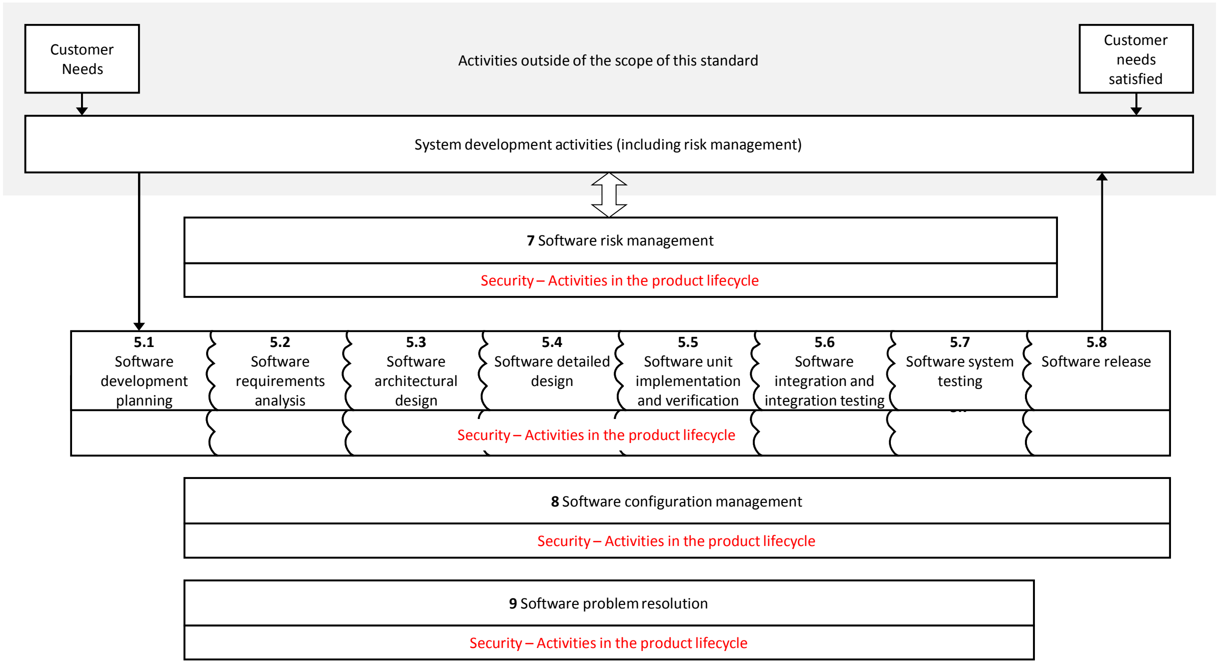
iss
INTEGRATED SCIENTIFIC SERVICES
A MEDTECH COMPANY

A quick look into the crystal ball

Cybersecurity – IEC CD 80001-5-1 Cyber Security Lifecycle (draft)

- Draft for IEC 80001-5-1 that would extend IEC 62304 software lifecycle processes with security related activities
- Current estimate for publishing is mid-2021

Cybersecurity – IEC CD 80001-5-1 Cyber Security Lifecycle (draft)


Customer Needs
Activities outside of the scope of this standard
Customer needs satisfied
System development activities (including risk management)
7 Software risk management
Security – Activities in the product lifecycle
5.1 Software development planning
5.2 Software requirements analysis
5.3 Software architectural design
5.4 Software detailed design
5.5 Software unit implementation and verification
5.6 Software integration and integration testing
5.7 Software system testing
5.8 Software release
Security – Activities in the product lifecycle
8 Software configuration management
Security – Activities in the product lifecycle
9 Software problem resolution
Security – Activities in the product lifecycle

INTEGRATED SCIENTIFIC SERVICES
A MEDTECH COMPANY

IEC 62304 Edition 2

- Second draft has been rejected too
- Current estimate for publishing is mid-2021

?