
# INTRODUCTION TO IEC 62304 MEDICAL DEVICE SOFTWARE

EPSMA, Nuremberg, Germany – 4th June 2018

Diarmuid Hogan, Manager Engineering

Excelsys Technologies, An Advanced Energy Company

# IEC62304

IEC 62304  covers Software Lifecycle process – it is not a Quality System!

IEC 62304 is very well laid out and easily readable – almost step-by-step guide to exactly what needs to be implemented and complied with.

This presentation not meant to be a substitute for an in-depth knowledge and implementation of IEC 62304 .

Special recognition of Phelim Bradley of Excelsys, and Liam Power of Reduktive Ltd

# RISK ASSESSMENT & SAFETY CLASSIFICATION

Determine the Class of your product or medical device according to Software Risk Management of IEC14971

- The safety classifications are
  - 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


This is the starting point of your process, all steps hereafter are determined by the safety classification

Where a hazard can arise from the failure of a software system to perform as specified, then the probability of failure can be assumed to be 100%!

# DOCUMENT DELIVERABLES

- Software Development & Maintenance Plan
- Risk Management File
- Software Requirements Specification
- Software Architecture Document
- Software Detailed Design Document
- Software Integration, Test and Validation
- Test Records, Change Control, Release Worksheets

# SOFTWARE DEVELOPMENT & MAINTENANCE PLAN

- Defines The Software Development Life Cycle Model
- Defines the Processes
- Defines the Deliverables
- Defines the Traceability
- Defines Configuration and Change Management Process
- Defines Problem Resolution

# SOFTWARE DEVELOPMENT & MAINTENANCE PROCESS

## Software Risk Management

- As discussed earlier

## Software Requirements Gathering

- Include risk control measures in software
- Revise and evaluate Device risk analysis and system requirements once software requirements are set
- Verify Software Requirements
  - Implement System Requirements
  - Ensure system and software requirements do not contradict
  - Avoid ambiguity
  - Testable
  - Uniquely identified
  - Traceable back to system requirements

# SOFTWARE DEVELOPMENT & MAINTENANCE PROCESS

**SRS contains**

- Functional and Capability Requirements
- System Inputs and Outputs
- Interfaces from Software System to other systems
- Alarms, warnings, messages, etc
- Security requirements
- Usability engineering requirements
- Data and database requirements
- Installation and Acceptance Requirements
- User docs to be developed
- User Maintenance requirements
- Regulatory requirements
- Requirements relating to operation and maintenance

# SOFTWARE DEVELOPMENT & MAINTENANCE PROCESS

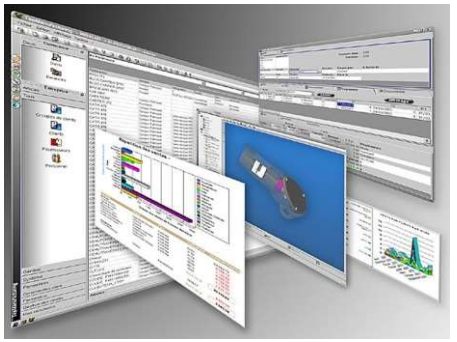
## Software Architectural Design

- Transform Software Requirements into architecture
- Develop an architecture for the interfaces between software items and hardware items
- Specify Requirements and proper functionality of SOUP items
- Identify partitioning for risk control

# SOFTWARE DEVELOPMENT & MAINTENANCE PROCESS

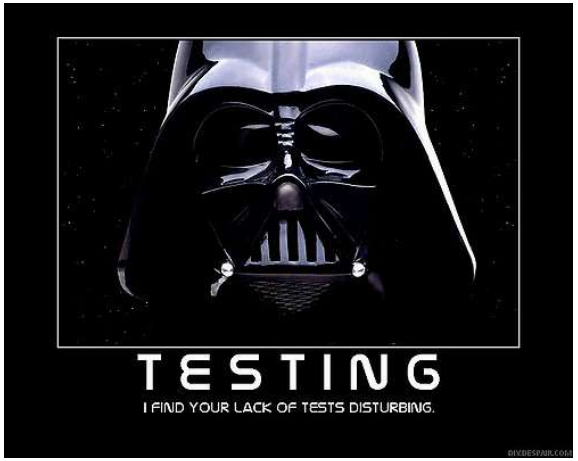
## Software Detailed Design

- Design and Implement each software unit
- Specify Test Strategy and procedure for each software unit
- Verify unit testing input and outputs are valid
- Establish acceptance criteria
- Perform verification, document results and store



## Software Integration and Testing

- Integrate software units in accordance with integration plan
- Test Integrated software
- Verify test process
- Perform regression testing



# SOFTWARE DEVELOPMENT & MAINTENANCE PROCESS

## Software System Testing

- Establish test for software requirements
- Retest after changes
- Verify Software System testing
- Record the tests (same as any hardware tests)
  - Record the Version under test
  - Document the Test Conditions and Results
  - Store the Result Records
  - Identify the Tester
  - Ensure sufficient info to allow test to be repeated

# SOFTWARE DEVELOPMENT & MAINTENANCE PROCESS

## Software Release

- Ensure software verification and validation is complete
- Document and evaluate known residual issues
- Document released versions
- Document the software design
- Ensure all open items are closed off
- Archive software
- Check Design for Manufacturing capability (Poke/Yoke)

# SOFTWARE DEVELOPMENT & MAINTENANCE PROCESS

## Software Maintenance

- Establish Software Maintenance Plan (very important!)
- Analysis when problem occurs and modification of code required
  - Monitor feedback
  - Document and evaluate
  - Evaluate effect on safety, if any
  - Need Software Problem Resolution process
  - Test, Verify and Validate modified software
  - Approve change request
  - Release modified software
  - Communicate to users and agencies where required
  - Process to update software in the field if required

# SOFTWARE DEVELOPMENT & MAINTENANCE PROCESS

## Software Configuration Management

- Identify all unique components of the Software system
- Change Control
  - Approve change request
  - Release modified software
  - Verify modified software
  - Provide audit trail for changes
- Use a Version Control system for check-in/check-out for all components of software system including hardware

# SOFTWARE DEVELOPMENT & MAINTENANCE PROCESS

## Software Problem Resolution


- Prepare problem reports
- Investigate
- Advise relevant parties
- Use change control process
- Maintain records
- Analyse problem for trends
- Verify software problem resolution
- Retain appropriate test documentation relating to changes

# SOFTWARE DEVELOPMENT & MAINTENANCE PROCESS

## Review

- Process to assess Software requirements – based on safety risk – Class C requires a lot more diligence and robustness and subsequently will result in much more code than Class A
- Process to Develop Software – released software must be traceable full circle back to SRS
- Process to Test Software – how, when, who and what. Store test data.
- Process to Maintain Software – once released how to handle changes that may need to be made to the software, internally and externally
- Be very cautious when using SOUP
- Important to explain to design team that the process should handle all the risks, not the individual!