ISO 25119-3:2018 (E)

Tractors and machinery for agriculture and forestry — Safety-related parts of control systems — Part 3: Series development, hardware and software

Contents

Foreword

Introduction

1 Scope

2 Normative references

3 Terms and definitions

4 Abbreviated terms

5 System design

5.1 Objectives
5.2 General
5.3 Prerequisites
5.4 Requirements
5.4.1 Structuring safety requirements
5.4.2 Technical safety concept
5.4.2.1 General requirements of technical safety concept
5.4.2.2 Specification of the technical safety concept
5.4.2.2.1 General
5.4.2.2.2 States and times
5.4.2.2.3 Safety architecture, interfaces and marginal conditions
5.5 Work products

6 Hardware

6.1 Objectives
6.2 General
6.3 Prerequisites
6.4 Requirements
6.5 Hardware categories
6.6 Work products

7 Software

7.1 Software development planning
7.1.1 Objectives
7.1.2 General
7.1.3 Prerequisites
7.1.4 Requirements
7.1.4.1 Phase determination
7.1.4.2 Process flexibility
7.1.4.3 Process timetable
7.1.4.4 Applicability
7.1.4.5 Supporting processes
7.1.4.6 Phases of software development
7.1.4.7 Using the tables
7.1.5 Work products
7.2 Software safety requirements specification
7.2.1 Objectives
7.2.2 General
7.2.3 Prerequisites
7.2.4 Requirements

7.2.4.1 Software safety requirements specification methods
7.2.4.1.1 Requirements specification in natural language
7.2.4.1.1.1 Aim
7.2.4.1.1.2 Description
7.2.4.1.2 Informal methods
7.2.4.1.2.1 Aim
7.2.4.1.2.2 Description
7.2.4.1.3 Semi-formal methods
7.2.4.1.3.1 Aim
7.2.4.1.3.2 Description
7.2.4.1.4 Formal methods
7.2.4.1.4.1 Aim
7.2.4.1.4.2 Description
7.2.4.1.5 Computer-aided specification tools
7.2.4.1.5.1 Aim
7.2.4.1.5.2 Description
7.2.4.2 Non-safety–related functions
7.2.4.3 Level of detail
7.2.4.4 Consistency
7.2.4.5 Hardware and software co-dependency
7.2.4.6 Software safety requirements specification
7.2.4.7 Software safety requirements verification
7.2.5 Work products
7.3 Software architecture design
7.3.1 Objectives
7.3.2 General
7.3.3 Prerequisites
7.3.4 Requirements
7.3.4.1 Software architecture design methods
7.3.4.2 Design method characteristics
7.3.4.3 Software architecture structure
7.3.4.4 Level of detail
7.3.4.5 Software architecture traceability
7.3.4.6 Software architecture verification
7.3.4.7 Combination of safety-related software components
7.3.5 Work products
7.4 Software component design and implementation
7.4.1 Objectives
7.4.2 General
7.4.3 Prerequisites
7.4.4 Requirements
7.4.4.1 Software component design and implementation methods
7.4.4.1.1 Suitable programming language
7.4.4.1.1.1 Aim
7.4.4.1.1.2 Description
7.4.4.1.2 Strongly typed programming language or guideline checking
7.4.4.1.2.1 Aim
7.4.4.1.2.2 Description
7.4.4.1.3 Language subset
7.4.4.1.3.1 Aim
7.4.4.1.3.2 Description
7.4.4.1.4 Tools and translators — Increased confidence from use
7.4.4.1.4.1 Aim
7.4.4.1.4.2 Description
7.4.4.1.5 Use of trusted/verified software components (if available)
7.4.4.1.5.1 Aim
7.4.4.1.5.2 Description
7.4.4.1.6 Defensive programming
7.4.4.1.6.1 Aim
7.4.4.1.6.2 Description
7.4.4.1.7 Structured programming
7.4.4.1.7.1 Aim
7.4.4.1.7.2 Description
7.4.4.1.8 Modular approach

7.4.4.1.8.1 Aim
7.4.4.1.8.2 Description
7.4.4.1.9 Complexity metrics
7.4.4.1.9.1 Aim
7.4.4.1.9.2 Description
7.4.4.1.10 Information hiding/encapsulation
7.4.4.1.10.1 Aim
7.4.4.1.10.2 Description
7.4.4.1.11 Library of trusted/verified software components
7.4.4.1.11.1 Aim
7.4.4.1.11.2 Description
7.4.4.1.12 Computer-aided design tools
7.4.4.1.12.1 Aim
7.4.4.1.12.2 Description
7.4.4.1.13 Use of coding standards
7.4.4.1.13.1 Aim
7.4.4.1.13.2 Description
7.4.4.1.14 Design and coding standards — No dynamic variables or objects
7.4.4.1.14.1 Aim
7.4.4.1.14.2 Description
7.4.4.1.15 Online checking during creation of dynamic variables or dynamic objects
7.4.4.1.15.1 Aim
7.4.4.1.15.2 Description
7.4.4.1.16 Design and coding standards — Limited use of interrupts
7.4.4.1.16.1 Aim
7.4.4.1.16.2 Description
7.4.4.1.17 Design and coding standards — Defined use of pointers
7.4.4.1.17.1 Aim
7.4.4.1.17.2 Description
7.4.4.1.18 Design and coding standards — Limited use of recursion
7.4.4.1.18.1 Aim
7.4.4.1.18.2 Description
7.4.4.2 Software component design and coding verification
7.4.5 Work products
7.5 Software component testing
7.5.1 Objectives
7.5.2 General
7.5.3 Prerequisites
7.5.4 Requirements
7.5.4.1 Software component testing methods
7.5.4.1.1 Boundary value analysis
7.5.4.1.1.1 Aim
7.5.4.1.1.2 Description
7.5.4.1.2 Checklists
7.5.4.1.2.1 Aim
7.5.4.1.2.2 Description
7.5.4.1.3 Static analysis — Control flow analysis
7.5.4.1.3.1 Aim
7.5.4.1.3.2 Description
7.5.4.1.4 Static analysis — Data flow analysis
7.5.4.1.4.1 Aim
7.5.4.1.4.2 Description
7.5.4.1.5 Dynamic analysis and testing
7.5.4.1.5.1 Aim
7.5.4.1.5.2 Description
7.5.4.1.6 Test case execution from boundary value analysis
7.5.4.1.6.1 Aim
7.5.4.1.6.2 Description
7.5.4.1.7 Structure-based testing
7.5.4.1.7.1 Aim
7.5.4.1.7.2 Description
7.5.4.1.8 Equivalence classes and input partition testing
7.5.4.1.8.1 Aim
7.5.4.1.8.2 Description

7.5.4.1.9 Test case execution from model-based test case generation
7.5.4.1.9.1 Aim
7.5.4.1.9.2 Description
7.5.4.1.10 Performance testing — Resource budget analysis
7.5.4.1.10.1 Aim
7.5.4.1.10.2 Description
7.5.4.1.11 Performance testing — Response time and memory constraints
7.5.4.1.11.1 Aim
7.5.4.1.11.2 Description
7.5.4.1.12 Performance testing — Performance requirements
7.5.4.1.12.1 Aim
7.5.4.1.12.2 Description
7.5.4.1.13 Performance testing — Avalanche/stress testing
7.5.4.1.13.1 Aim
7.5.4.1.13.2 Description
7.5.4.1.14 Interface testing
7.5.4.1.14.1 Aim
7.5.4.1.14.2 Description
7.5.4.2 Elimination of defects
7.5.5 Work products
7.6 Software integration and testing
7.6.1 Objectives
7.6.2 General
7.6.3 Prerequisites
7.6.4 Requirements
7.6.4.1 Software integration and test plan
7.6.4.2 Software integration strategy
7.6.4.3 Software integration test procedures
7.6.4.4 Software integration test methods
7.6.4.4.1 Functional testing
7.6.4.4.1.1 Aim
7.6.4.4.1.2 Description
7.6.4.5 Elimination of defects
7.6.5 Work products
7.7 Software safety testing
7.7.1 Objectives
7.7.2 General
7.7.3 Prerequisites
7.7.4 Requirements
7.7.4.1 Software safety testing methods
7.7.4.1.1 Tests within the electronic control unit network
7.7.4.1.2 Hardware-in-the-loop tests
7.7.4.1.3 Tests in the machine
7.7.4.2 Extent of tests
7.7.4.3 Software safety requirements validation
7.7.4.4 Documentation
7.7.4.5 Elimination of defects
7.7.5 Work products
7.8 Software-based parameterisation
7.8.1 Objective
7.8.2 General
7.8.3 Prerequisites
7.8.4 Requirements
7.8.4.1 Data integrity
7.8.4.2 Executable code in parameter data
7.8.4.3 Configuration management
7.8.4.4 Software-based parameterisation verification
7.8.5 Work products

Annex A (informative) Example of agenda for assessment of functional safety at AgPL = e

Annex B (normative) Independence by software partitioning

B.1 Overview
B.2 Terms, definitions and abbreviated terms

B.2.1 Terms and definitons
B.2.2 Abbreviated terms
B.3 Objectives
B.4 General
B.5 Requirements
B.5.1 General requirements
B.5.1.1 SRL
B.5.1.2 Software architecture
B.5.2 Several partitions within a single microcontroller
B.5.2.1 General
B.5.2.2 Software partitioning methods/measures
B.5.2.3 Software partitioning effectiveness
B.5.3 Several partitions within the scope of a micro-controller network
B.5.3.1 General
B.5.3.2 Methods for multi-processor partitioning
B.5.3.3 Multi-processor partitioning effectiveness

Page count: 59