SIL2LinuxMP Linux Qualification - Process Overview

Nicholas Mc Guire <safety@osadl.org>

January 25, 2016

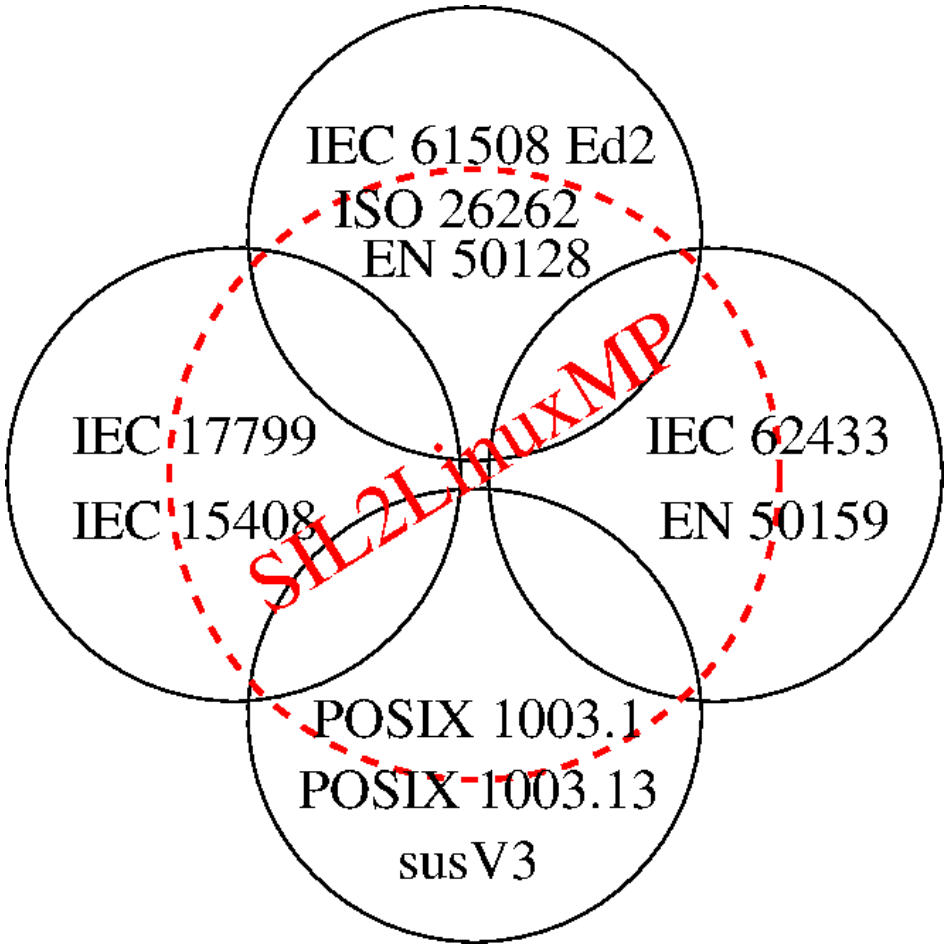
Outline

- Context
- Process
- Conclusions

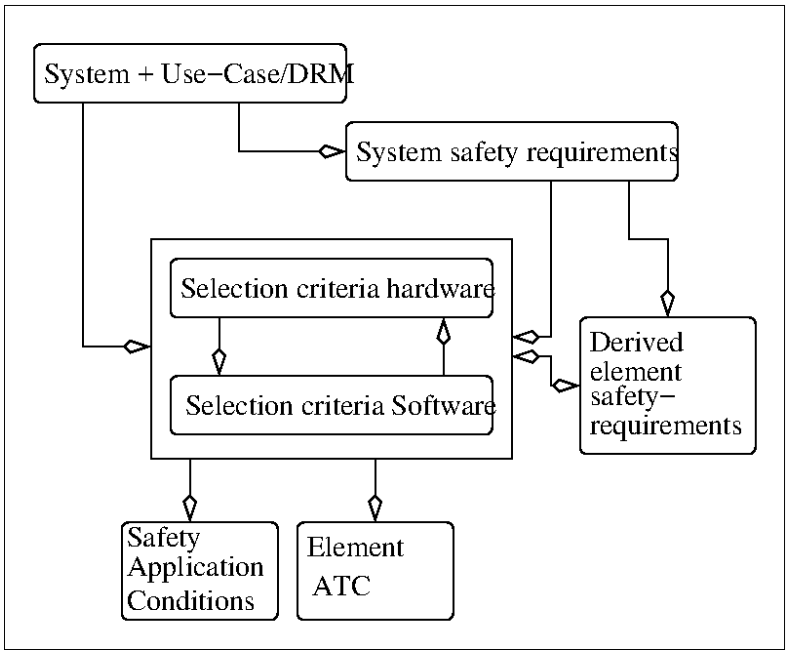
Goal of SIL2LinuxMP

- Generic qualification approach
- Suitable for up to SIL2 (IEC 61508 Ed 2)
- Support multicore systems
- Mainline kernel + glibc + tools
- Methods suitable for pre-existing SW intensive systems

SIL2LinuxMP Context



SIL2LinuxMP Selection



Selection has been formalized in the context of 61508-1 Ed 2 as Clause 7.X "E/E/PE safety-related software element selection" - pennding review by TueV Rheinland.

3_S Asessment of non-compliant development

```
7.4.2.12 +- a) Route S 3
         |      '-> Compliance to 7.4.2.13
         |           +- a) adequate software safety requirements specification
         |           |     '- 7.2 safety functional capability/integrity
         |           +- b) safety properties satisfy
         |           |     +- 7.2.2 -> 7.2.2.2 -> 7.4.2.12 (loop TODO -> CA)
         |           |     +- 7.4.3 architecture design
         |           |     +- 7.4.4 tools and languages
         |           |     +- 7.4.5 software system design
         |           |     +- 7.4.6 code implementation
         |           |     +- 7.4.7 software module testing
         |           |     +- 7.5 HW/SW integration
         |           |     +- 7.7 system safety validation
         |           |     +- 7.8 software modification
         |           |     +- 7.9 software verifiation
         |           |     '- 8 functional safety assessment
         |           +- c) element documentation (functional and SC)
         |           |     +- 7.4.3 architecture design
         |           |     +- 7.4.5 software system design
         |           |     '- 7.4.6 code implementation
         |           +- d) evidence requirements for software integration
         |           +- e) evidence of systematic V\&V
         |           |     +- 7.4.7 software module testing
         |           |     +- 7.4.8 software integration testing
         |           |     +- 7.5 HW/SW integration
         |           |     +- 7.7 system safety validation
         |           |     '- 7.9 software verifiation
         |           +- f) evidence of non-interference by unused functions
         |           +- g) credible failure mechanisms identified and mitigated
         |           |     +- 7.2.2.4 Assessment of independence
         |           |     +- -1 7.3 Hazard scope - contributions by environment
```

3_S Asessment of non-compliant development - cont.

```
|           |     `- -1 7.4 Hazard and risk analysis
|           +- h) identification of build and runtime environment
|           |     +- 7.3.2.2 g) Credible failure mechanisms identified
|           `- i) valid only for applications complying with safety manual
`- b) Safety Manual
       +-> 61508-2 Annex D (see 61508-3 D.2.1)
       `-> 61508-3 Annex D
            +- D.1 Purpose -- Documentation of
            |   +- D.1.1 functions, constraints and evidence),
            |   +- D.1.2 is to be created during system design,
            |   `- D.1.3 all user relevant attributes for deployment.
            +- D.2 Content of safety manual includes
            |   +- D.2.1 all relevant parts of 61508-2 Annex D
            |   +- D.2.2 unique identification and deployment instructions
            |   +- D.2.3 element configuration of SW<->HW and assumptions
            |   `- D.2.4 integrator competence, element compliance,
            |            compatibility and limitations, interface needs.
            `- D.3 Justification of claims
                +- D.3.1 supporting evidence for all claims
                |   `- 61508-2 7.4.9.7 (bug loops back to Annex D)
                +- D.3.2 non-selfreferencial safety manual :)
                `- D.3.3 restates 61508-2 7.4.9.7 NOTE 2
```

Adjusted software DLC

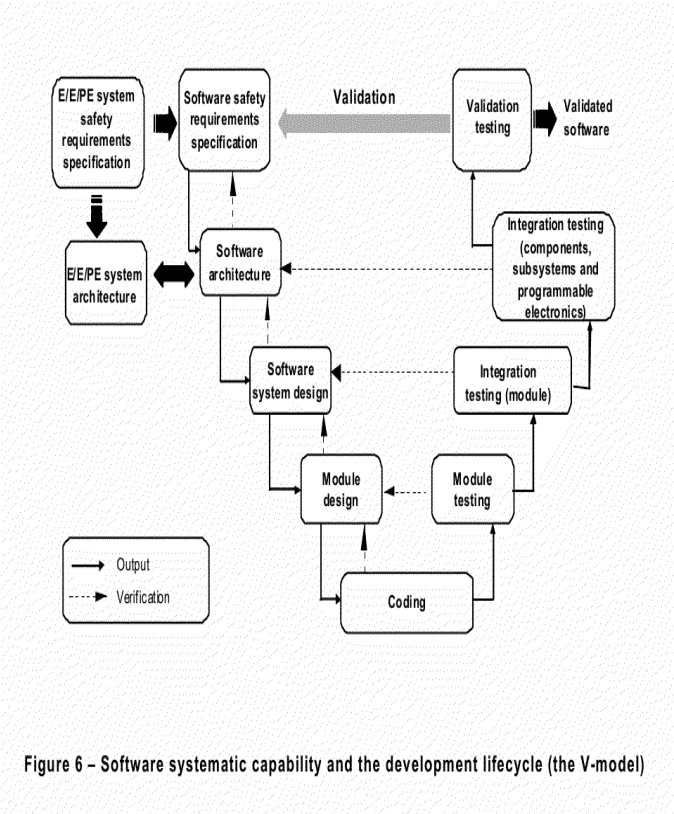
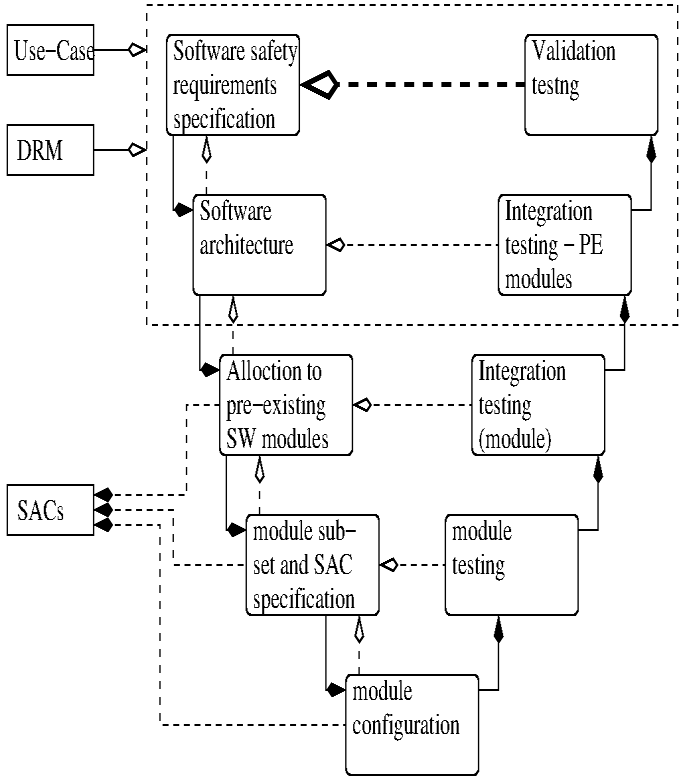
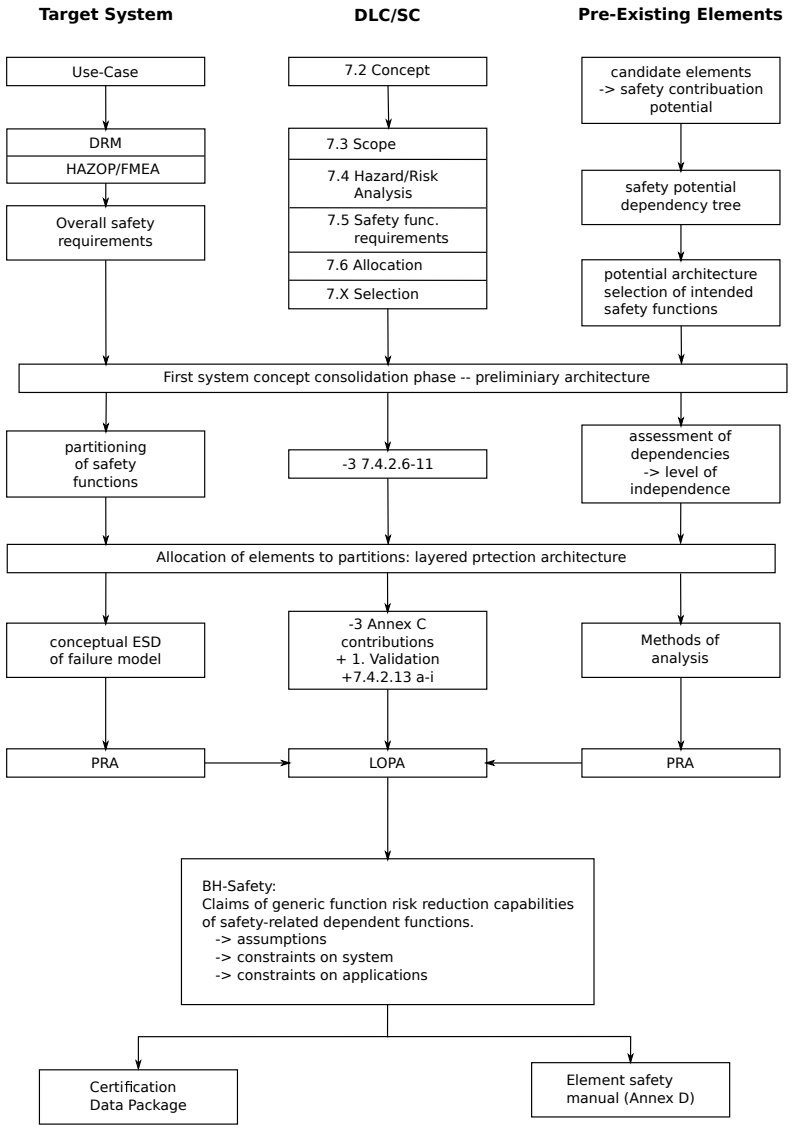


Figure 6 – Software systematic capability and the development lifecycle (the V-model)



Softwar systematic capability – V–model for pre–existing softwa

Big picture of DLC/SLC



Conclusions

- If you want to utilize FLOSS -> fix the processes first
- ISO 26262 is **not** really usable for software intensive systems
- IEC 61508 was not really conceived with selection as primary strategy in mind - but it **is** doable.
- The process adjustments are in review (TueV Rheinland) ... lets see
- Based on the final processes the method set will be selected
- Applying this to GNU/Linux RTOS will not be trivial - but looks doable

We will report on progress along the way...