ELISA Work Group – Qualification of Linux software development, deployment and maintenance process

ENABLING LINUX IN SAFETY APPLICATIONS

Dec 2019/Jan 2020

WG goal

- The Linux development, maintenance and deployment process itself is well established by now and provides a solid foundation for our work.
  - The goal of the WG is two-fold:Define a **reference process** which can be shown to be as effective (equivalent in assurance) as legacy models (as described in safety standards such as ISO26262 or IEC61508).
  - Demonstrate that the **Linux process is equivalent** to this reference process.
- In doing so, we expect to identify gaps in the current process.
- For each such gap, we can either provide mitigations which are sufficient in particular domains; or work within the community in order to promote change.

ENABLING LINUX IN SAFETY APPLICATIONS

An Intel Company

Linux Assessment

- **Goal:** Analyze the complete Linux design, development, and release process against the reference process
- **Outcome:** Assess the current development flow with predictability of releases and test completeness in mind
- **Results:** Proposals to Linux foundation about how to improve the project management and development activities, AOUs for integration in safety systems. For example, an AOU might be to run stress tests on specific platforms.

ENABLING LINUX IN SAFETY APPLICATIONS

An Intel Company

Proposed plan to qualify Linux process

- The **reference process** shall cover the following areas
    - Design
    - Development
    - Verification
    - Infrastructure
    - Maintenance
    - General
- The reference process shall be aligned with safety standard expectations
- The reference process shall be used to demonstrate the equivalence of the Linux process

ELISA
ENABLING LINUX IN SAFETY APPLICATIONS

Process Area: Design

- **Requirements definition:** Collection process, traceability.
- **Software architecture** – readiness for safety can define
  - ➢ Good design principles to ensure properties such as cohesion, interoperability, modularity, etc.
  - ➢ Timing and execution management
  - ➢ Resource management

  NOTE: Given that Linux is far too complex as a single unit, we focus on subsystems. We organize subsystems into logical groups based on relevant use cases (safety critical embedded systems)
- **SW implementation/coding** – Provision of style and coding guidelines in accordance with safety standard expectations
- **Design review:** Provision of guidelines to perform design review and to measure the quality of source code against pre-defined "static" targets (e.g. complexity checking)

ENABLING LINUX IN SAFETY APPLICATIONS

An Intel Company

Process Design: Development

- **Project management** including planning for safety activities; how to appoint roles & responsibilities
- **Version-control system** – focus on tools (e.g., GIT for code management)
- **Release planning** – based on short release process.
    - ➢ Developer branches and architecture branches are supported for early testing of features
    - ➢ Well defined experimental tree and merge window
    - ➢ Early testing of release candidates.

ENABLING LINUX IN SAFETY APPLICATIONS

An Intel Company

Process Area: Verification

- **Testing hierarchy and testing process**, difference steps for validation
    - ➢ Linux test project
    - ➢ Test frameworks such as kernel self test
    - ➢ Continuous integration process managed by distro owners (e.g., RedHat)
- **Kernel testing** –
    - ➢ How the kernel is tested (e.g., https://kselftest.wiki.kernel.org/)
    - ➢ How non-compliances are treated

Legacy model references both white-box as well as black-box testing.

The Linux software verification processes will be analyzed against both types of testing measures.

ENABLING LINUX IN SAFETY APPLICATIONS

Process Design: Infrastructure

- **Change management** – management of different change types (defects, anomalies, modification of safety features)
- **Configuration management** – e.g., https://elixir.bootlin.com/linux/v5.4/source
- **Tool chain** – selection of tools that can (or must?) be used during the different process steps (from specification to integration and testing..)
- **Build process,** including commands, scripts and dependencies to generate the final software
- **Acceptance criteria for software release -** Acceptance level of the final build e.g., zero errors vs zero warnings
-

ELISA
ENABLING LINUX IN SAFETY APPLICATIONS

An Intel Company

Process Design: Maintenance

- **Impact analysis for new release** (new features, changes, patches) –
  - ➢ Linux development follows a distributed, hierarchical model, with over 100 subsystems developed independently and each maintained by an expert.
  - ➢ WG should analyze for safety impact all data on patches, reviews, bugs; as well as the design, coding, review and release process for new features.
- **Retention policy** – definition of retention policy aligned with domain applications.

  For example, after EOL (End-of-Life), for how many years must documents / evidences / code be retained?

ENABLING LINUX IN SAFETY APPLICATIONS

An Intel Company

Process Design: Maintenance(con’t)

- Identification of **safety related patches or changes or new features** (our assessment will identify gaps for this point).
    - ➢ *Existing features to be qualified* – e.g., bad block management by ubifs; futext contention handling; safe file system mount.
    - ➢ *New features to be proposed* – e.g., dynamic page protection; safe Linux boot.
- **Consensus-oriented model**, with strong enforcement and review by the development community. This is fundamental, and the strengths of this model as compared to legacy development processes need to be analyzed for safety.
- **High-level management aspects**, including annual kernel summit for strategic decisions and domain specific groups.

ENABLING LINUX IN SAFETY APPLICATIONS

An Intel Company

Process Design: General

- The following characteristics (derived from ISO 26262-6§74.1 and IEC 61508-3§7.4.3) shall be addressed:
  - ➢ Simplicity and understandability
  - ➢ Consistency and completeness with respect to software safety requirements specification
  - ➢ Verifiability and testability
  - ➢ Modularity, abstraction and encapsulation
  - ➢ Integration Information
  - ➢ Maintainability

Example – Safe Linux boot

- Linux boot is configurable, scalable, maintainable, testable. *Leverage this ability to hard-code a safe kernel configuration into the Linux kernel build.*
- Define safety requirements on the Linux kernel image configuration: *what needs to be configured.* For example, cgroups for resource management can be used to define:
  - Minimum CPU resources allocated to ensure fair scheduling.
  - Which process may create devices as well as open them for read and/or write.
  - Performance monitoring of a set of processes in a cgroup
- *Resource management impacts system safety. Appropriate configuration can ensure the kernel will safely provide fair and reliable resources for all system process.*

ENABLING LINUX IN SAFETY APPLICATIONS

An Intel Company

Systemd Unit Files – use case

- Features:
  - ➢ Users, groups, access control
  - ➢ Capabilities
  - ➢ Resource limits
  - ➢ Device management, file system safety
  - ➢ Network configuration
  - ➢ Application lifecycle
- Qualification, “bridging the gap”
- Mapping safety requirements to the technology

ELISA
ENABLING LINUX IN SAFETY APPLICATIONS

MOBILEYE An Intel Company

Safe Linux Boot (con’t)

- Based on the safety requirements, we can derive a reference implementation: *how it will be configured.* For example, a reference implementation of systemd unit files.
- The reference implementation can only be tested in the context of a specific business use case. Specific boot configuration requirements are defined for the use case: *what configuration settings will be set.*
- Feature enhancements may be added to the reference implementation by other organizations. These enhancements need to follow a well-defined test and release process.

ENABLING LINUX IN SAFETY APPLICATIONS

Tasks

- **Design:**
  - ➢ Requirements definition
  - ➢ SW architecture readiness for safety - Elana
  - ➢ SW implementation/coding guidelines - Joy
  - ➢ Design review guidelines
- **Development:**
  - ➢ Project management
  - ➢ Version control
  - ➢ Release planning - Lukas
- **Verification:**
  - ➢ Test hierarchy and test process - Elana
  - ➢ Kernel testing

Tasks (con't)

- **Infrastructure:**
  - ➢ Change management
  - ➢ Configuration management
  - ➢ Tool chain
  - ➢ Build process
  - ➢ Acceptance criteria for software release
- **Maintenance:**
  - ➢ Impact analysis for new releases
  - ➢ Retention policy
  - ➢ New features, patches - Elana
  - ➢ Consensus-oriented model and how it fits into our reference model
  - ➢ Management aspects

ELISA
ENABLING LINUX IN SAFETY APPLICATIONS

MOBILEYE An Intel Company

Template – task report

What we should provide for each task, ~ 10-15 pages (*we need WG consensus on this)*:

- A clear **definition of the task** and its scope. 1-2 paragraphs.
- A clear (laymen's terminology) breakdown of **goals and deliverables** of the task including a graphical description of the flow and how it fits into our reference process model.
- **Mapping** of the task on to ISO26262 requirements.
- **Background research** on existing tools, frameworks, solutions.
- **Gaps** identified and how it is proposed to fill those gaps (mitigations, patches, or otherwise).
- **Technical background and references**.

Workshop, Brussels – Jan 30/31

- Goal is to include a technical session for our WG
    - ➢ Milestone – Finalize work plan, define “owners” for each task before the Workshop
- Tentative agenda for WG session:
    - ➢ Come prepared to the workshop with my 3 tasks to be discussed with group
    - ➢ Collect intermediate results from owners of all additional tasks
    - ➢ Move forward on “reference process”
    - ➢ Set milestones in time for each future workshop
- Post workshop:
    - ➢ Agreed reference process
    - ➢ Measures of “equivalence” (aka “good enough”)
    - ➢ Gaps and resolutions

ELISA
ENABLING LINUX IN SAFETY APPLICATIONS

An Intel Company