



Isar Community Meetup | 8.12.2025 | Nürnberg

Using ISAR to hold, remix and distribute Apt Snapshots via Git

Andreas Naumann | emlix GmbH

emlix dates & figures



- Founded 2000
- 65+ employees, Organic growth, project-driven organization
- Quality and information security: ISO 9001:2015, TISAX® AL3
- License and security compliance: IEC 5230:2020, IEC 18974:2023
- Core competence: development and maintenance of industrial grade embedded Linux systems and applications
- Customer Industry: Medical, Automotive, Machine Industry, Industrial Automation, Electrical Design and Board Manufacturers
- Me: Andreas Naumann, Dipl.-Ing.
 - System Engineer Embedded Linux
 - Electronics, BSP, Buildsystems, CI & Testautomation

Partner & Project

Our Partner:

- Elektrobit Automotive GmbH
 - market expertise in automotive
 - background in SPICE and functional safety
 - established award-winning and visionary global vendor of embedded and connected software products and services for the automotive industry

Project:

- EB corbos Linux for Safety Applications (EBcLfSA)
 - new approach to run SIL2 safety applications directly on a Linux system
 - the first OS-solution assessed by TÜV Nord to be compliant with safety standards up to SIL2/ASILB and based on embedded Linux.



Some Context

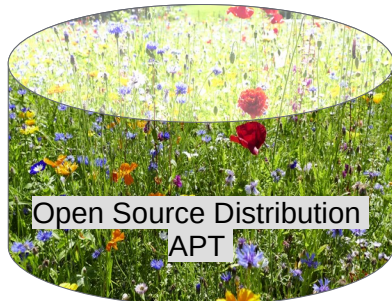
The Job

- Develop, integrate and co-maintain a buildsystem,
- which allows for distribution and customization of the EbcLfSA product
- for the next 10-15 years
- (and solve some problems with the previous solutions)

Challenge 1: Diverse Ecosystems

Combine Software Components from different ecosystems into a sustainable product tailored distribution, e.g.

- Debian derived Open Source Distribution
- EB/emlix safety stack, consisting of hypervisor, supervisor, 1-n Linux-VMs, supporting Userspace
- Yet unknown 3rd party components + end customer SW



Challenge 2: EBcLfSA Context

→ Finding a solution for Linux for Safety Applications means:

”finding a solution within the one of the most diverse and inhomogeneous engineering context one can think of”

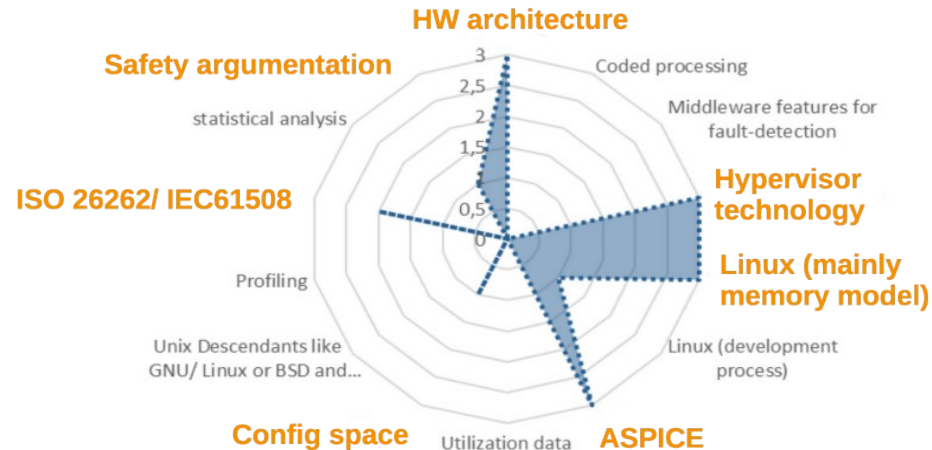
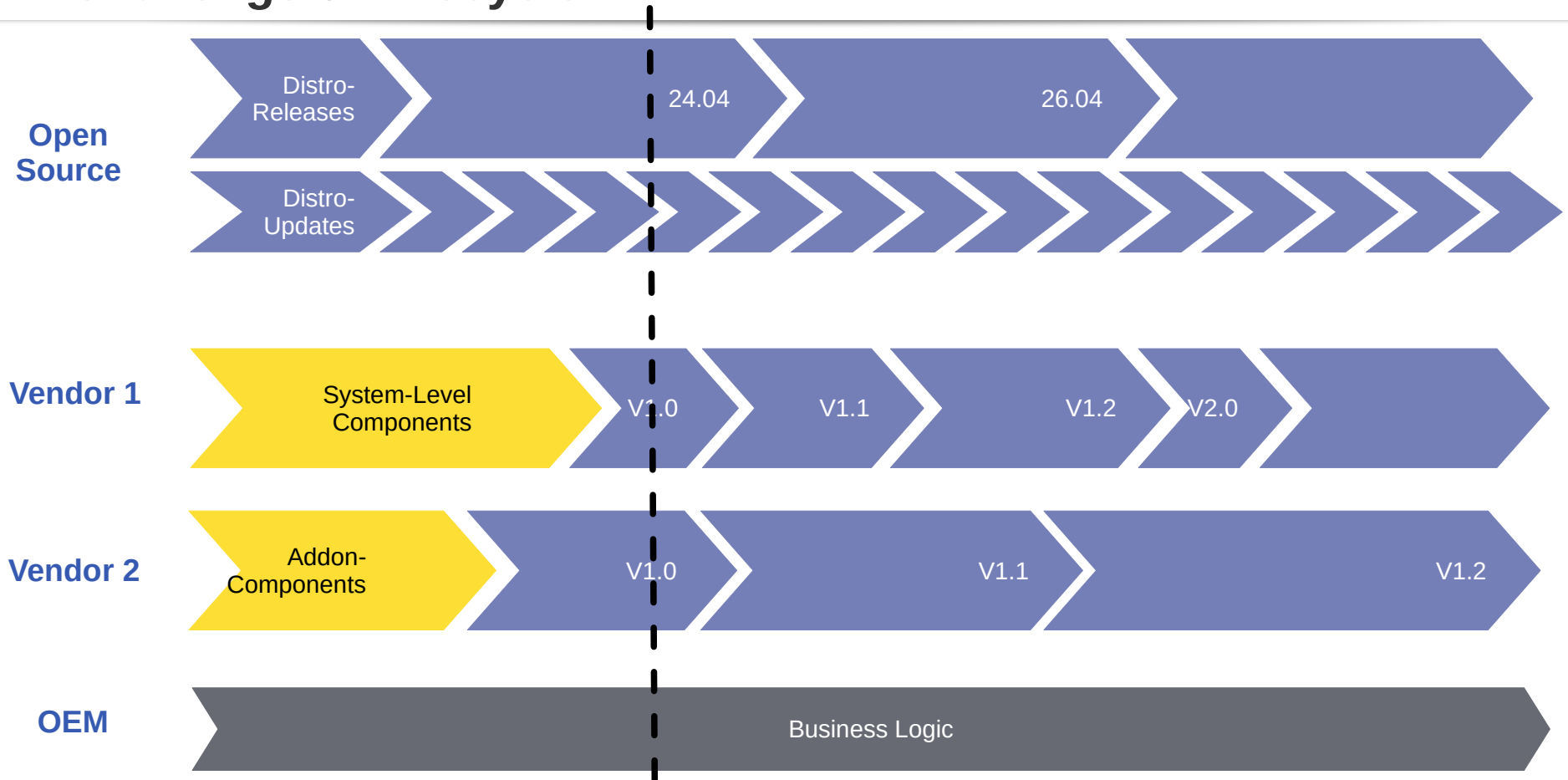













diagram derived from M. Ambruster, F. Arrighetti (2024), “Linux for safety-related applications”. exida Automotive Symposium, Spitzing, 2024.

Challenge 3: Lifecycle



Looking for a suiteable Buildsystem...

... and found ISAR to tick most boxes:

- compatible with the Debian/Ubuntu ecosystem 
- with bells and whistles for Embedded 
- works on Debian packages
 - prebuilt packages 
 - rebuild upstream packages 
 - build custom packages 
- reproducible builds 
- defines the composition 
- build without network  
- avoid redundant recompilation  



Core Requirements

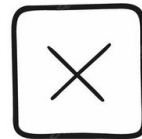
Package Composition

Core Requirement I:

A finegranular, one file per package, definition of

- available packages
- and their meta data:
 - version
 - dependencies & provides
 - original URL
 - hashes
 - ...

shall be tracked/secured in Git and independently/locally auditable.



Build without Network

Core Requirement II:








Allow for local build without network access (archived .deb files only)

- ISAR_USE_CACHED_BASE_REPO ☐ (first run still needs external apt)
- no way to get „minbase“ package set to isar-apt/ ☐
- bootstrap doesn't run from pre-populated isar-apt/ ☐

Recompile binaries on demand

Core Requirement III:

Build packages from sources only if not available from prebuilt Ubuntu or previous build (custom).







- reuse of upstream debian packages (dpkg-prebuilt) 
- reuse of custom package builds
 - with sstate:
 - nice in CI 
 - touchy during development 
 - easy transfer over organization boundaries 
 - with external apt  
 - without sstate and without external apt 

emlix/EB Solution for the Requirements I+II

(git tracked composition; no network during build)

- *Reposync* tool to **generate dpkg_prebuilt recipes** for a given APT (configurable by suite, components, architectures, package whitelist)
- add (multiarch scoped) **dependencies** on the generated recipes to
 - isar-mmdebstrap-host|target
 - sbuild-chroot-host|target
 - image/sdk
- **support bootstrapping** from prepopulated isar-apt/

emlix/EB OS Contributions

- ISAR:
 - Speedup dpkg-prebuilt 
 - Optional „Rules-Requires-Root“ 
 - Enable bootstrapping on isar-apt/ (patch send, rework needed) 
 - Fixes/Tests for handling of transitive dependencies (which we have a lot and complicated examples of) 
 - Skip incompatible multiarch extension of architecture-specific recipes 
- Reposync:
 - Roadmap on Opensourcing pending 

Solution Concept For Requirement III

(Build and Distribute Release Packages w/o APT)

- src: Release-Build custom dpkg recipe (with Isar Revision+BuildID)
- bin: Reposync the resulting (QM approved) packages from isar-apt to designated prebuilt layer (commit+tag as needed)
- archive .deb Artefacts to Fileserver/Mirror
- publish the prebuilt layer and fileserver to eligible parties

Needed for development and CI:

- automatic switch between src/bin recipe, based on (dependency-) version bump
- idea: Use PRSERV to provide incremental package revisions for use in the Isar Revision+BuildID (and ultimately the prebuilt recipe version).



Scenarios

Hold: Snapshot Debian package metadata to Git

Snapshot \triangleq a run of *reposync* + *git commit*:

- generates Bitbake dpkg-prebuilt recipes (currently ~1600)
- freezes the metadata at that point in time to Git

Advantages:

- Git tracking, transparency and integrity
- ubiquitous tooling
 - e.g. `git log <path-to-package>/`
 - cherry-pick, merge/rebase (conflicts), revert, ...
- Pull request and merge verification workflows (auditable) for the composition

Ubuntu Layer:

```
meta-corbos-ubuntu-packages/
├── conf
│   └── layer.conf
├── recipes-admin
│   ├── adduser
│   │   └── adduser_3.137ubuntu1.bb
│   ├── amd64-microcode
│   │   └── amd64-microcode_3.20231019.1ubuntu2.1.bb
│   ├── ansible
│   │   └── ansible_9.2.0+dfsg-0ubuntu5.bb
│   ├── ansible-core
│   │   └── ansible-core_2.16.3-0ubuntu2.bb
│   ├── apt
│   │   ├── apt_2.7.14build2.bb
│   │   ├── apt_2.8.3.bb
│   │   ├── apt-utils_2.7.14build2.bb
│   │   └── apt-utils_2.8.3.bb
│   ├── arch-test
│   │   └── arch-test_0.21-1.bb
│   ├── base-files
│   │   └── base-files_13ubuntu10.bb
│   ├── base-passwd
│   │   └── base-passwd_3.6.3build1.bb
│   ...
```

Hold: Snapshot Debian package metadata to Git

further potential:

- Bitbake „knows“
- use Git/Bitbake to assert control
 - modify single recipe (needs locking)
 - bbappend (or better BBMASK)
- Reposync: prepare audit information for commit messages

Demo Build

e.g. libparted2t64_3.6-4build1.bb

```
SUMMARY = "disk partition manipulator - shared library"
DESCRIPTION = ""
SECTION = "libs"
SOURCE_PACKAGE = "parted"
HOMEPAGE = "https://www.gnu.org/software/parted"

inherit dpkg_prebuilt

SRC_URI = " <mirror>/pool/dev/p/parted/libparted2t64_3.6-4build1_amd64.deb;name=deb0 \
          <mirror>/pool/dev/p/parted/libparted2t64_3.6-4build1_arm64.deb;name=deb1 \
          "

SRC_URI[deb0.sha256sum] = "24e84d557b5037d1ad984ba1b6e5136c56..."
SRC_URI[deb1.sha256sum] = "e497519ac5610358b1184f26bfff520615..."

RDEPENDS = " \
    libblkid1 \
    libc6 \
    libdevmapper1.02.1 \
    libuuid1"
RDEPENDS:append:amd64 = " \
    dmidecode"

RPROVIDES = " \
    libparted \
    libparted2"
```

Mix: multiple package sources and custom software

- kas setup of Bitbake layers with
 - generated recipes
 - and custom recipes
- e.g. add repo with generated recipes from PPA in kas instead of `DISTRO_APT_SOURCES += ...` (which is actually)
- Reposync can take on maintenance tasks, e.g.
 - consistency check on dependencies (are they available in any layer?, are there duplicates? Do they fulfil version requirements, are there multiple versions of the same package? ...)
=> uncover possible problems early (within snapshot/commit activity)
 - Download/Archive the corresponding .deb files

Redistribute

- only respective Git-Repo and Deb-Files needed (Fileserver, NFS, archive)
- storage in apt repository optional
- integrity and traceability is ensured by bitbake/lisar
- with CI Support (build and archive debs if needed)
 - allows reproduction of images entirely from binary packages
 - for all states of development: feature, maintenance, customer variant branches
- still vision: Seamless switchover to recompile/package in case source code or config packages were changed



Perspective

Ideas...

- Automating DEBIAN_DEPENDS → DEPENDS
- Automating rootfs DEPENDS for „isar-apt only“ use case
 - atm: function to convert debian package to multiarch Isar recipe names
- Less privileges (first step switch kas-container to CAP_SYSADMIN?)
- What's the difference between base-apt and bootstrapping from isar-apt?
- Public layer with generated Debian/Ubuntu Recipes
- Carry prebuilt recipes for bootstrap required packages (could solve downgrade issues for custom APT dependencies)

Wishes...

- know when the next release will be cut
- more feedback on patches/RFCs
- patience with debian newbees

Questions ?



How can we support you?

emlix GmbH
Göttingen | Berlin | Bonn | München

Headquarters
Berliner Str. 12
D-37073 Göttingen / Germany

Fon +49 (0) 551 / 306 64 - 0
solutions@emlix.com
www.emlix.com

