

EMBEDDED ENTWICKLUNG LEBENSZYKLUS

Was alles zur Embedded Programmierung gehört



Christoph Sterz
christoph.sterz@kdab.com

Hi
Ich bin Christoph



About Me

- Bachelor & Master am **HPI** (zwischenndrin **Cern**, Genf)
- Masterarbeit am OSM Lehrstuhl: ~ Linux perf auf NUMA
- Währenddessen schon bei **KDAB** gearbeitet
 - C++/Qt Firma in Berlin
 - >70% unserer Mitarbeiter sind OpenSource Devs
- Heute mach ich Linux perf(ormance) auf Embedded*
* und mehr :)



Master's Thesis
**Analyzing NUMA Performance
Based on Hardware Event Counters**
Christoph Sterz
July 22, 2016

Warum halte ich heute diesen Vortrag?

- Embedded Entwicklung & Consulting ist mein Beruf geworden.
 - Consulting = (sehr schnell sehr viel Code lesen || Gerätesoftware verstehen);
 - Consulting != Viel Reden
- Ich denke, dass ich einen guten Einblick habe:
 - **Infotainment:** 3 Auto SW-Plattformen, Yachts für Scheichs, Flugzeuge
 - **Medizingeräte:** (Beatmungsgeräte, Mikroskope, Autoklaven, Zellzähler)
 - **Industrie:** (SCADA-PLCs, Finnische Sägewerke, Weltklasse-Ampere-Eichgeräte)
 - *Gerade* **Braumaschinen** von Hobby mit Stecker (10L/20L) bis zum Ratskeller (1000L, 60KW CAN Heizungen & Frequenzgeber)



Warum halte ich heute diesen Vortrag?

- Oft dreht es um Produkte / Produktlinien die **realweltliche Anforderungen** haben*
- **Embedded Operating Systeme** und das Gesamte System werden darauf hin angepasst und bietet der eigentlichen Software notwendige Funktionen an
 - Klappt manchmal gut, oft muss man “anbauen”.

* Wir reden heute viel darüber was die Industrie alles für sie wichtiges “erreichen will/muss” und was man dafür tun muss.

Disclaimer

- Das ist alles meine eigene kleine Weltsicht
- Kein Vollständigkeitsanspruch
- Es werden Produkt und Firmennamen fallen
 - Es existieren andere Produkte und Firmen die auch gut oder besser sind :)

Es gibt Bonuspunkte für gute Zwischenfragen!!



Produktlebenszyklus

Entwicklungs-Phase 0: Bevor es losgeht

Zuallererst: Zertifizierungen

- Bevor alles beginnt: Produkte müssen legal sein
- Zertifizierungen und ISOs bestimmen die Industrie in ihren Planungen

Es können gute und schlechte System-Architekturen entworfen werden aber an der Zulassung kommt niemand vorbei.

- ISO 26262 “FuSi” (Sicherheitslevel ASIL A-D)
- ISO 13458 *Medical Devices – Quality management systems*
- Für Hardware: EMV Prüfung, Flugzeug HW ~ 10G Beschleunigungstests

Hardwareplanung first, Software second

- Hardware ist sehr oft preisbestimmend
- Software folgt dem, was vorhanden ist
- Automotive HeadUnit+InstrumentCluster: 4-8 Cores Arm + GPU
 - Höchste Stückzahlen
- Industriegerät: 1–2 Cores ARM, GPU optional
 - Hohe Stückzahlen
 - Powerhorse der Industrie NXP IMX.6 (“Raspberry Pi der Industrie”)
- Medical Gerät: Ab einem gewissen Preislevel COTS x86 (Intel/AMD)
 - Oft niedrige Stückzahlen

Realistische Zahlen für Projekte

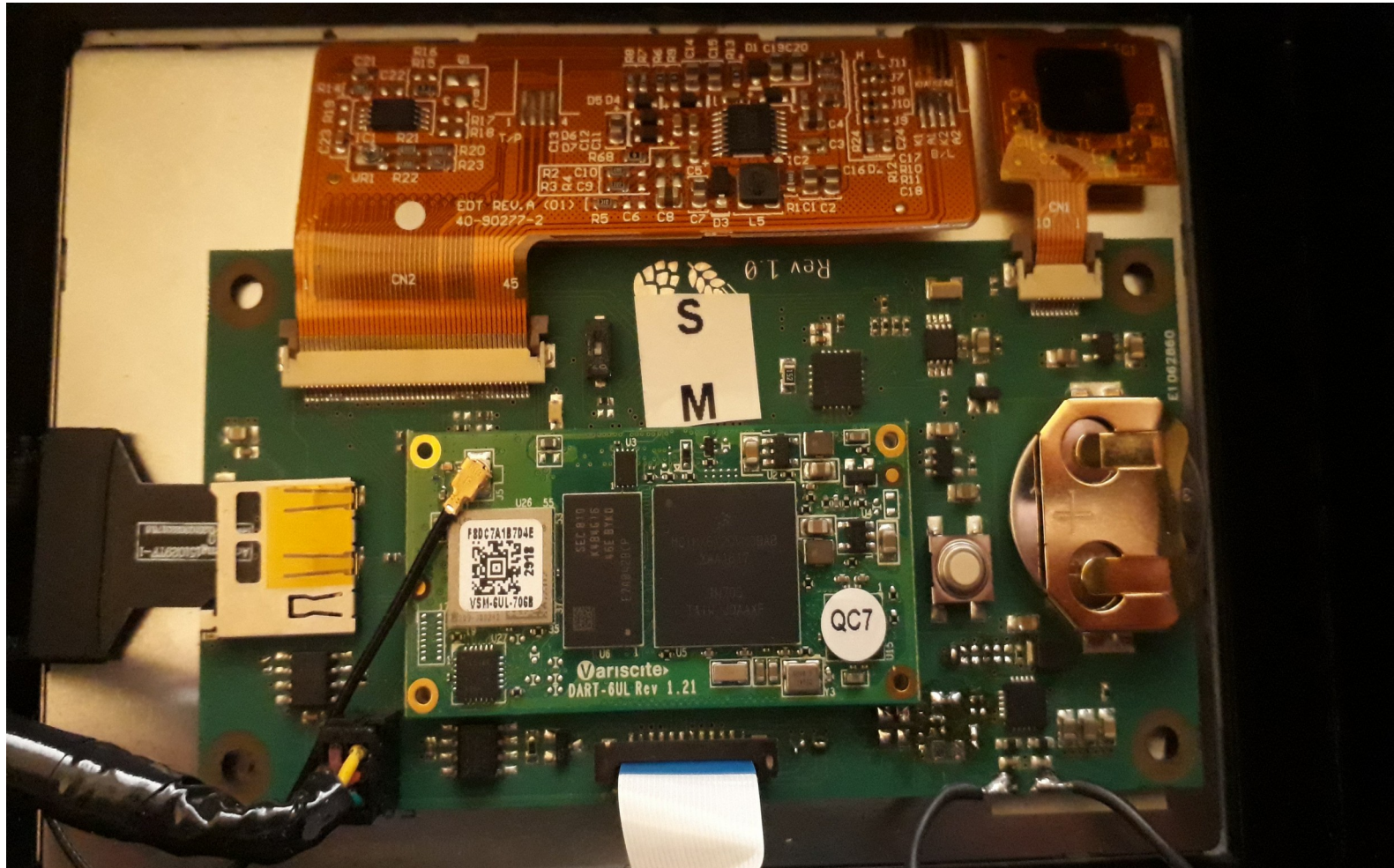
- Hausgeräte: bis 30Mio.
- Autos: bis 40 Mio.
- Industrie-Appliances: 20.000
- Industrie Kleinserien: 3.000
- Medizingeräte: 500, 5.000, Patientenbezogen mehr möglich

1GB mehr Ram kosten ~10\$ – Auswirkungen auf die Beispiele

Eval Board



SOM – System on a Module



Kundenanforderungen

- Always On
- Personalization
- Full Internationalization (I18n)
 - Nicht nur live Übersetzungen, Left-To-Right, Right-To-Left Oriented Design
- Updates
 - Over The Air
 - Targeted Updates / Canary Updates
- Reliability
 - A/B Images, Fallbacks, Werkseinstellungen...
- Smartphone ist der Standard gegen den verglichen wird.

Produktlebenszyklus

Entwicklungs-Phase1: Zu wenig

Prototypen

Eine Tatsache. Viele Auswirkungen...

- In meiner Erfahrung gibt es zuerst keine dann wenig Hardware.
 - Automotive-Entwicklungszyklen können 15 Jahre dauern
 - Mein letztes Auto-Projekt war mit 5 Jahren extrem schnell
 - Beta-Chips für Entwicklung, damit aktuell im SOP (Start of Production)
- Performance: Desktop/Simulatoren führen zu Täuschung
- Verfügbarkeit: 2-Klassen Entwickler mit und ohne Zugang zu Hardware
- Wenig Testing / Wenig Autotesting
- Viel Kupferlackdraht auf den wenigen Prototypen

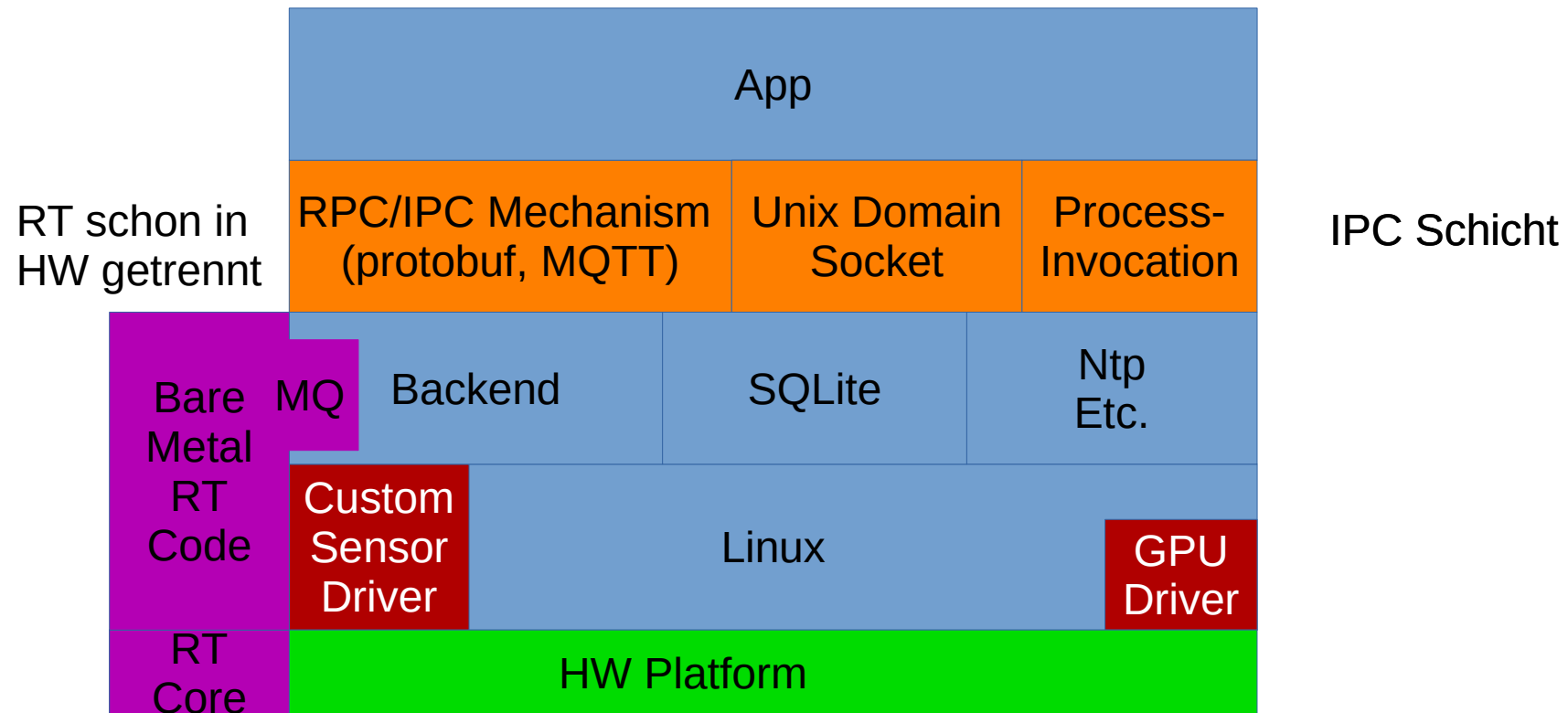
Opa erzählt vom Krieg (I)

- Autoprojekt, >10Mio Autos, A-Silicon (=CPU-Beta)
- Random Crashes in App immer in Function Calls.
- Nachgeschaut: Crashes auch in allen möglichen anderen Systemdiensten
- 2 Monate Debugging
 - Ist es libc?
 - Ist es der Crashhandler, der falsches reportet?
 - Wem kann ich noch trauen?
 - Soll ich vielleicht irgendetwas mit Holz machen, statt mit Software?
- Dann: Crashdumps: Immer nach einer gewissen Instruction (STP)
 - Hardware Bug, Thermal Issue in ARM ISA StorePaired, μ -code Patch nötig

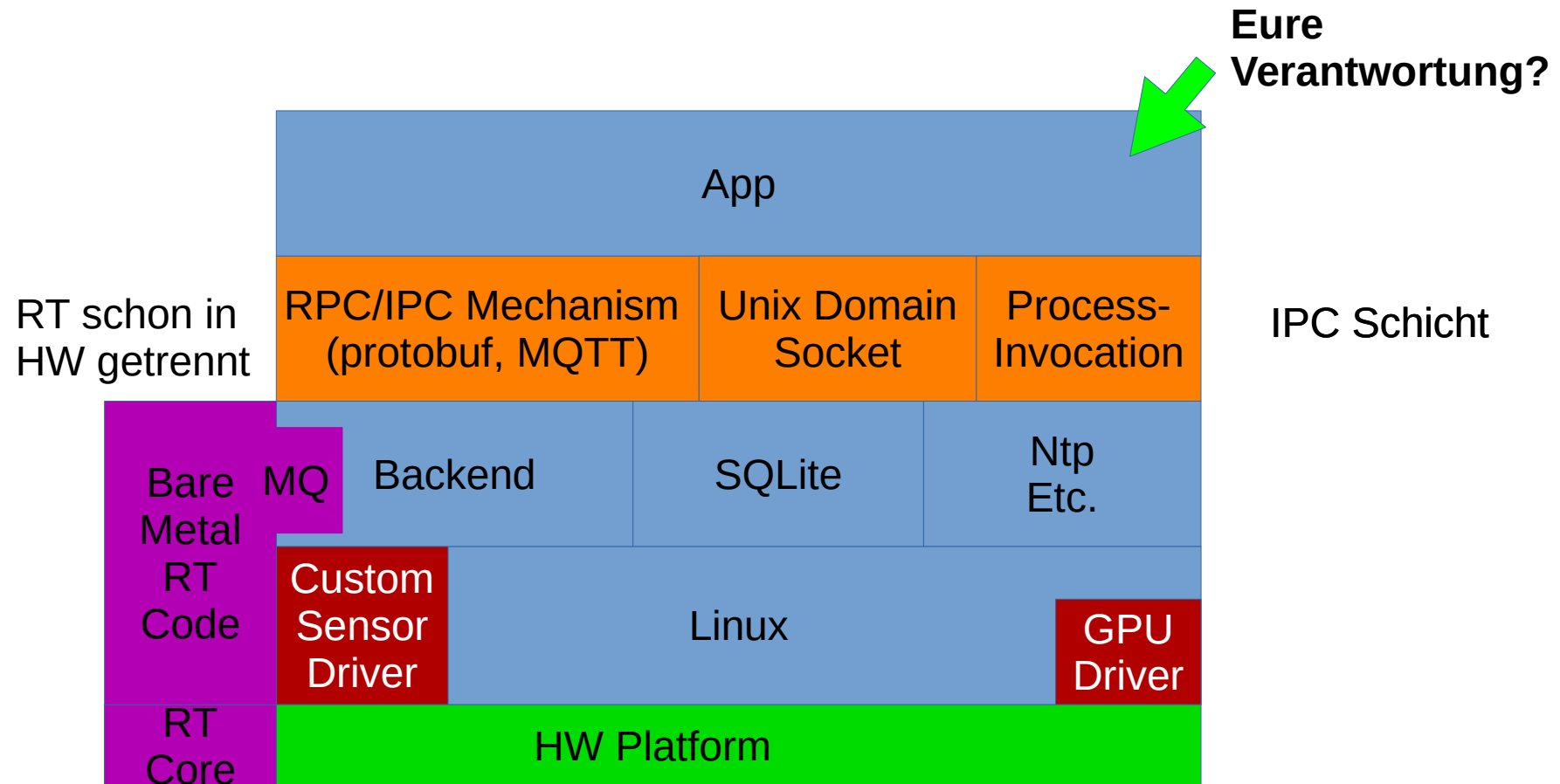
Software-Stack Typische Architektur

- **Linux+Android**: 70%, Bare-Metal: 10%, Windows: 10%, QNX:5%, VXWorks/Integrity: 3%, iOS...
 - Windows stirbt aus, EndOfLife für WinCE lang vorbei, kein RT mehr
- Linux: **Yocto**/Buildroot: 70%, Distro (oft Debian-*): 30%
- Linux: **Systemd**: 80%, SystemV: 20%
- Linux: **DBus** Support: >50%
- Code: Treiber: C, Business Logic **C++**, Python, C#
 - Bisher wenig Java außerhalb von Android gesehen
- GUI: **Qt**: >60% ← Observation Bias!
- IOT everywhere: Hersteller-API
- (schlimm) Altes Webkit mit HTML Usermanual

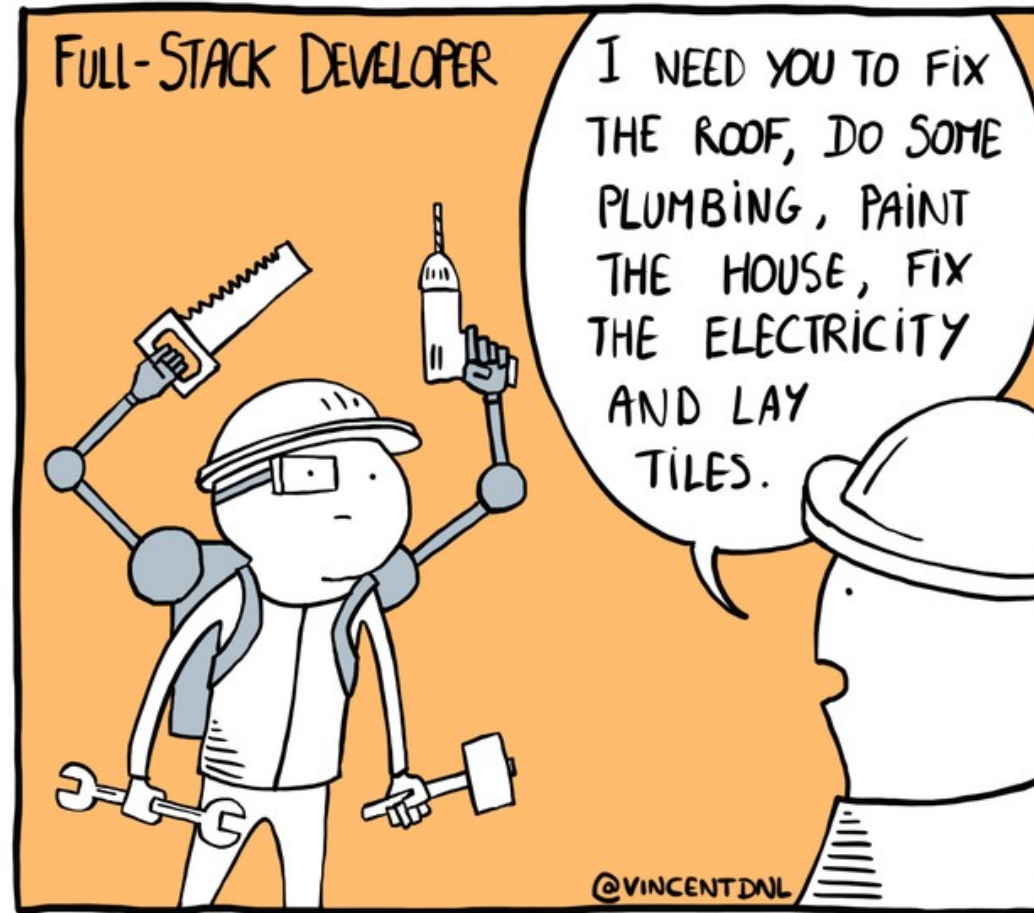
Architekturen “Der Klassiker”



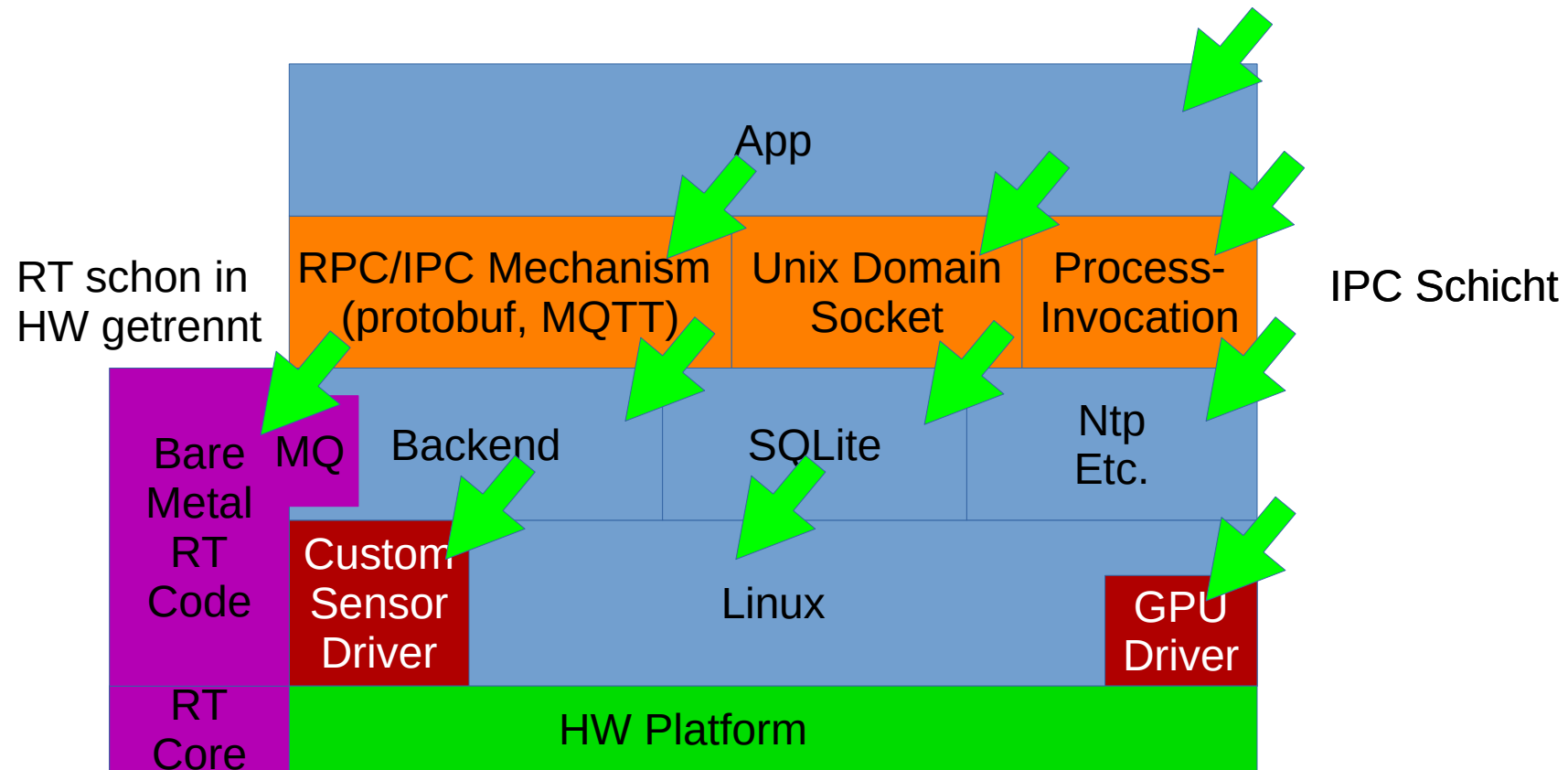
Architekturen “Der Klassiker”



Von der Hardware bis zum eigenen Code



Architekturen “Der Klassiker”



Opa erzählt vom Krieg (II)

- Prototypen laufen super
- Vorserienproduktion: 150 Devices, viel hellere Displays werden verbaut
- **Anruf:** alle Displays bleiben Schwarz
- **Kunde->Panik**
- SW Fehler: Linux GPIO Driver setzt **GPIO-direction** auf "**out**"
 - **!?** intial auf **LOW**
- Nächste Code-Zeile setzt sie auf **HIGH**
- Neue Displays hatten anderes timing, 100ns haben den Unterschied gemacht
- Lösung: SW-Patch zur korrekten *Direction* "**high**" (es gibt: in, out, low, high)

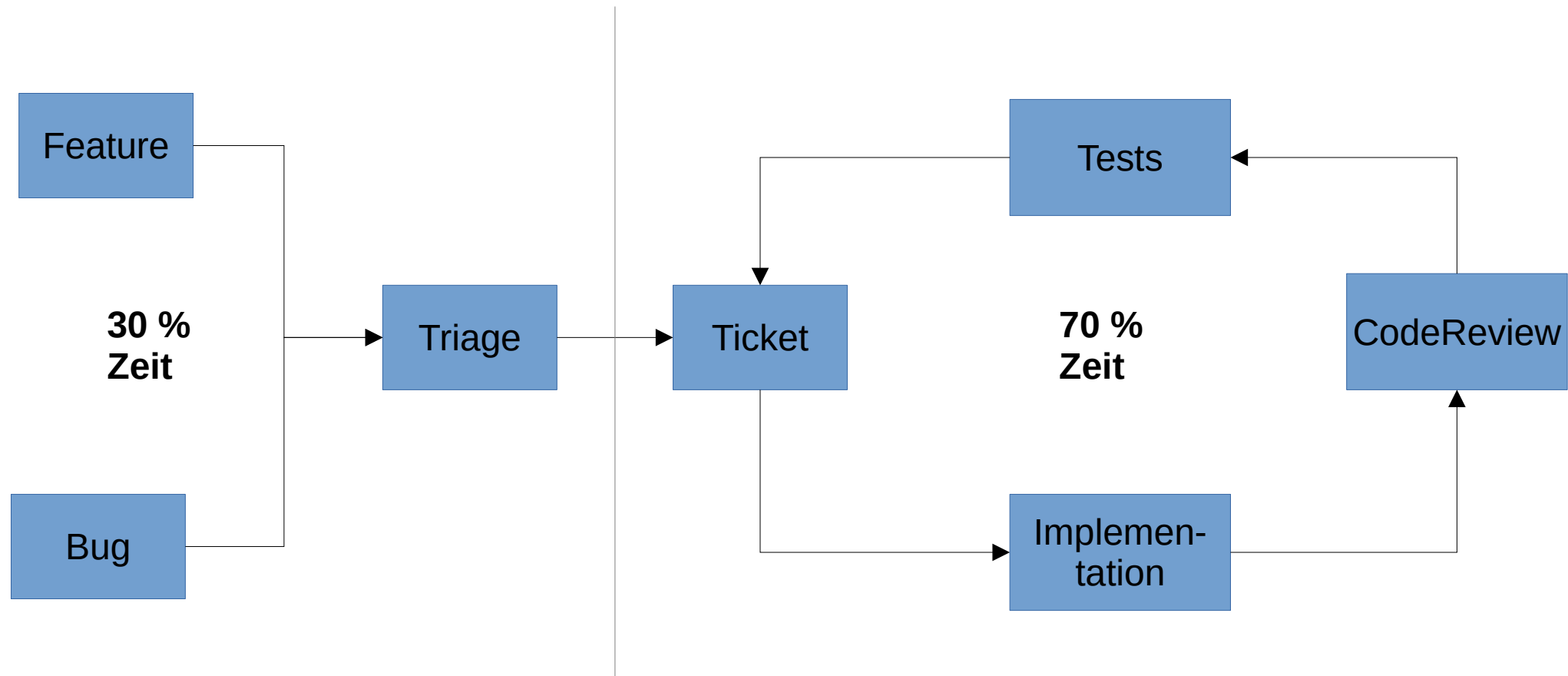
Produktlebenszyklus

Entwicklungs-Phase2: Kunde sieht Produkt zum 1. Mal

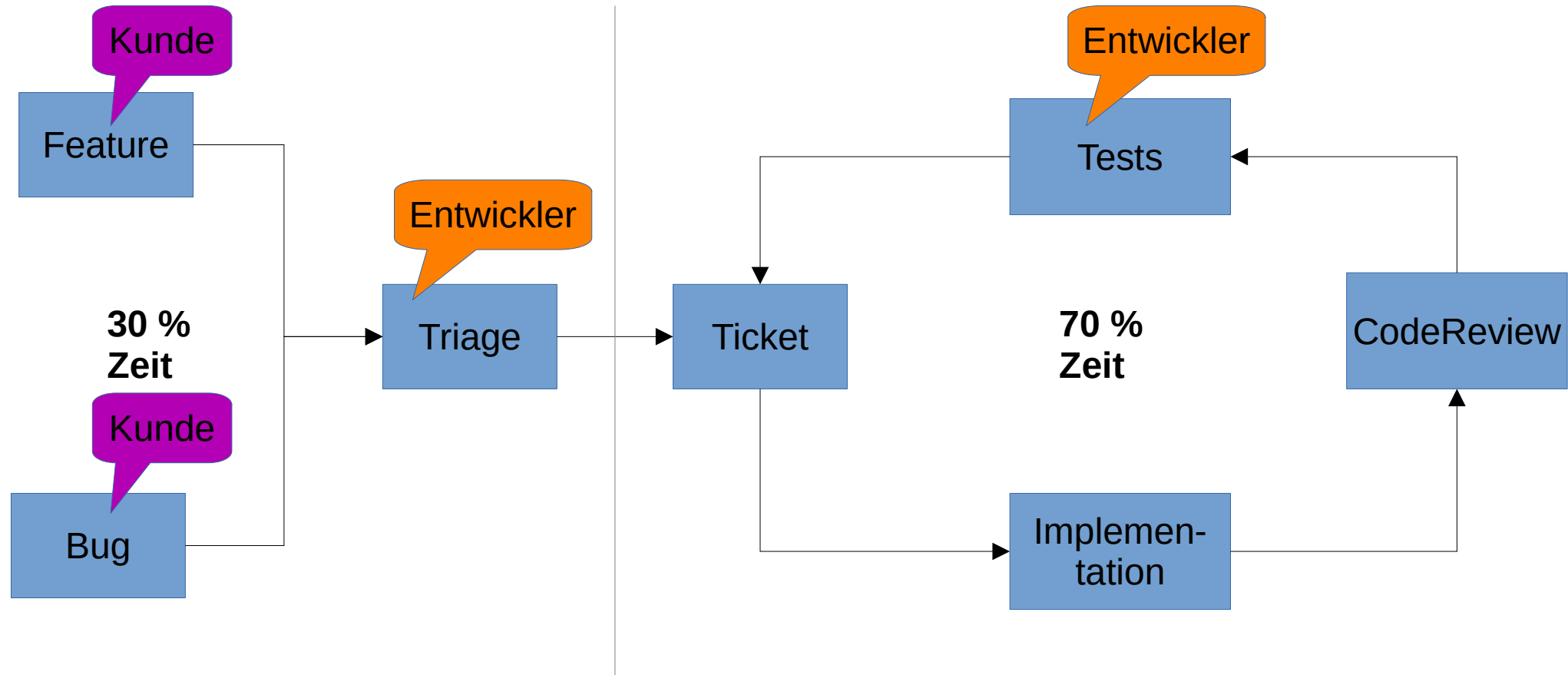
Alles ändert sich

- Ungeachtet des Entwicklungsmodells (selbst bei Wasserfall) kommt jetzt ein harter Bruch
 - Kunde testet und sieht Wünsche erfüllt / nicht erfüllt.
 - Designänderungen, UX Änderungen, Business Logic Änderungen
 - Definitionen werden klar, weil konkret
 - Kunde wird vor harte Entscheidungen gestellt, die bisher vage blieben (out of Spec)
- UI/UX: Die Expectations sind immer Smartphone-Grade
 - Selbst in der Industrie
 - **Frage:** Wieviele Entwickler haben bei Nokia an der virtuellen Tastatur gearbeitet?
 - **Frage:** Wieviele Entwickler arbeiten an der virtuellen Tastatur für euren Kunden?

Entwicklungsalltag



Entwicklungsalltag



Entwicklungsaltag OS: Yocto Project

- Distro-Erstellungs Baukasten
- Basierend auf HW-Board-Support-Packages
- Sich überlagernde Schichten von Rezept-Layern
 - Basisrezept
 - Erweiterung
 - Eigene Erweiterung
- Kompiliert erst den Cross-Compiler, dann Linux Images, dann SDK und Updates.




```
# for coredumps and better debugging
#INHIBIT_PACKAGE_STRIP = "1"
#INHIBIT_PACKAGE_DEBUG_SPLIT= "1"

DEPENDS += " \
    qtbase \
    qtserialbus \
    qtdeclarative \
    qtmultimedia \
    qtconnectivity \
"

RDEPENDS_app += " \
    iw \
    wpa-supPLICANT \
    i2c-tools \
    mosquitto \
    alsa-utils \
    alsa-state \
    qtmultimedia \
    qtmultimedia-plugins \
    iproute2 \
    can-utils \
    tzdata \
    fonts \
    canopenSocket \
"

FILES_${PN} += " \
    /opt/app/bin/app \
    /opt/backend/bin/backend \
    /opt/wifidaemon/bin/wifidaemon \
    /opt/mosquitto/ \
    /opt/mosquitto/access.acl \
    /opt/mosquitto/passwords \
    /opt/wifidaemon/wpa_backup.conf \
    /opt/app/sounds/click.wav \
    /opt/app/sounds/notify.wav \
    /opt/app/sounds/notify_boosted10db.wav \
    /opt/WebBridge/bin/WebBridge \
    /opt/testrig/bin/testrig \

```

Produktlebenszyklus Entwicklungs-Phase3: Start of Production SOP

SOP naht – Die Stunde der Performanceprobleme

- Jetzt kulminieren die Features
- Große, separat entwickelte Teile kommen auf der Hardware zusammen
- Es gibt endlich genug Prototypen, es wird auf echter Hardware getestet
- Zeit wird knapp
- Auch Stunde der Quick & Dirty Hacks

Erfahrung: 5 Jahre Performance Consulting

- Premature Optimization is the Root of all Evil

Bevor man versucht besonders smart zu optimieren, sollte man aufhören an anderer Stelle Performance zu verschwenden

- Die Probleme sind oft an unerwarteten Stellen
- Hardware verhält sich auf dem Gerät anders
- “Death by 1000 Cuts”
- Computer sind schnell!

Die Klassiker

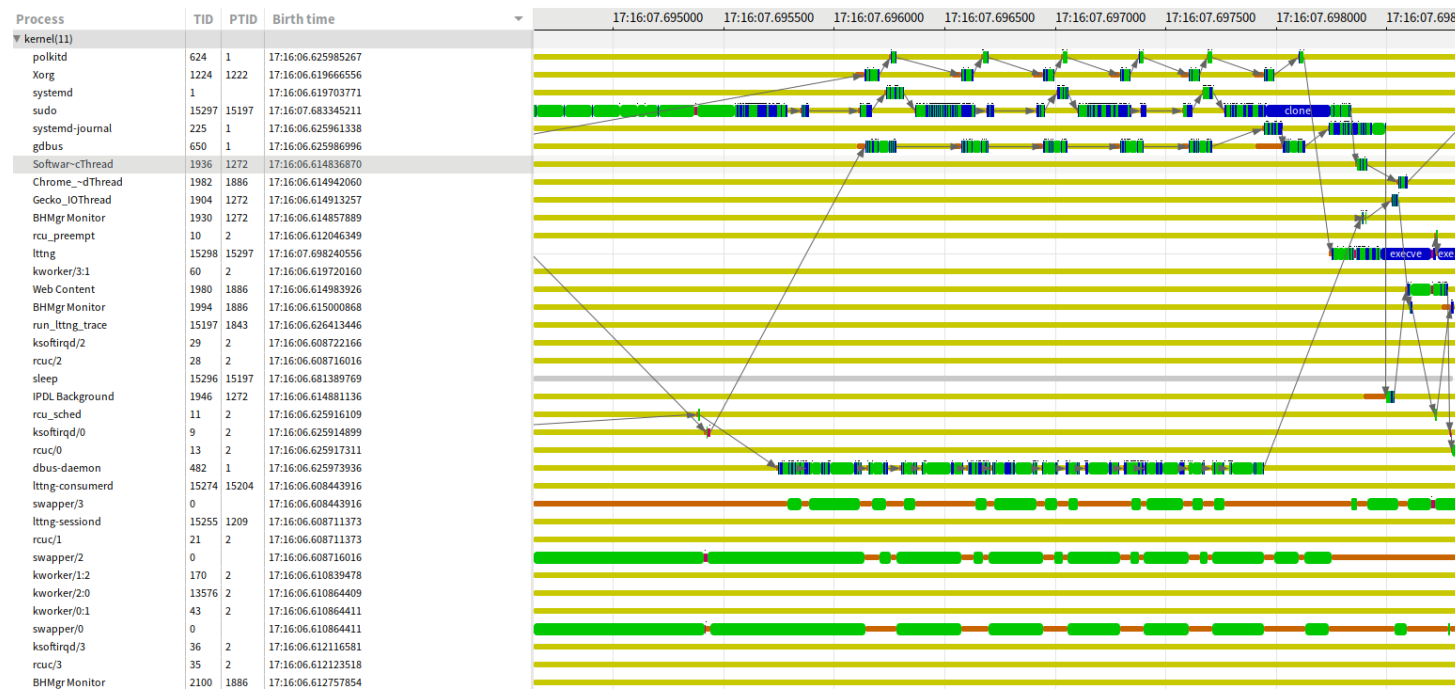
- “Warum ist da ein sleep?”
- Helfen, Datenmengen einschätzen zu lernen
 - Hochoptimierte Suchfunktion für 200 Listeneinträge à 1KB
 - Aber falsches ImageFormat im 12MB Background-Bild
 - Pfad der Daten(Mengen) folgen, Kopien beseitigen
- Jeden hochfrequenten Timer In Frage Stellen
- Heap-Allozierte Objekte in einer tight Loop
 - Falls nicht tcmalloc (etc.) verwendet wird: Der Heap ist Singleton
 - Global Heap Lock.

Das Werkzeug #1: Linux Perf & Flamegraphs

- Erwähnte ich Linux Perf :) ?
 - Low Overhead
 - Stack Unwinding auf dem Desktoprechner möglich
- **Beispiel** Hotspot (open Source perf viewer)
- Dann mit Entwicklern über die Stellen in der großen CodeBase Reden
 - “Ihr macht in eurer Renderloop DB-Anfragen, soll das so?”
 - “Jedes Logging Hat einen Timestamp, Jeder Timestamp fragt beim OS die Zeitzonen-Tabelle an, soll das so?”
 - “Ah! Nein, das war so nicht geplant.” Momente abwarten

Sonstige Alltägliche Werkzeuge

- Strace | heaptrack | die compiler sanitizer
- Linux Tracing Tools next Generation (LTTNG) für Systemweites tracing
 - Bei den anderen OS Herstellern gibts oft Herstellertools, die sind oft schlechter



Produktlebenszyklus

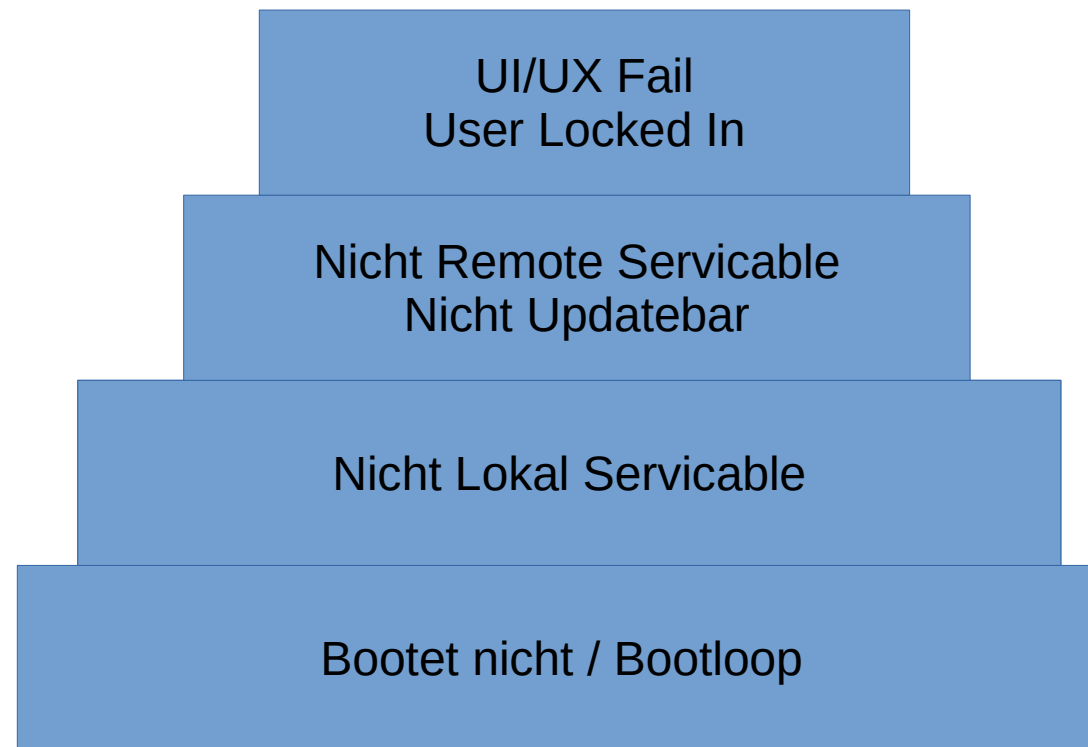
Entwicklungs-Phase4:

Erste Hardware Beim Endkunden

Ängste

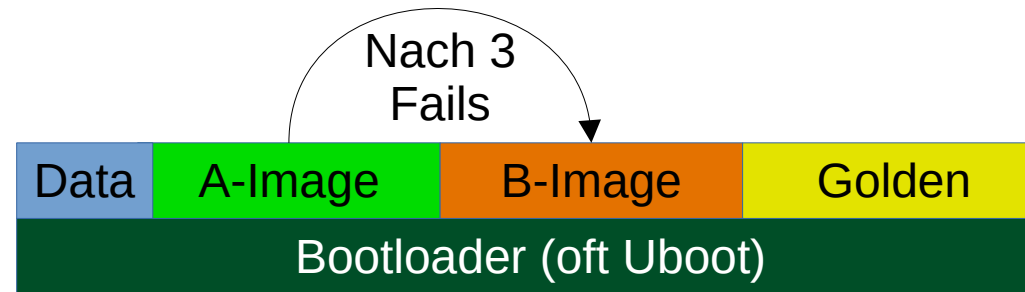


Gerät wird zum “Brick”



Strategien gegen Bricks

- Zur Entwicklungszeit: Immer eine UART einfordern!
- A / B Image
- “Golden” Image



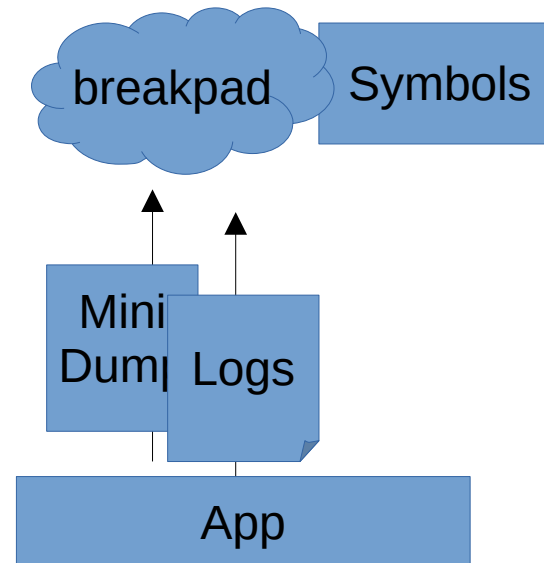
- Wifi /Connection Access Manager als separater Prozess!
- Werkseinstellungen
- Website-Flasher a la FritzBox

Updates

- Canary Updates:
 - Zielgerichtet auf TestNutzer / Normale Nutzer
 - Spezialversionen für einzelKunden
- 2 Grundlegende Ansätze:
 - SW-Update – Gesamtes Image wird geflashed
 - OSTree – Diff basiert “git fürs Filesystem”
- Datengröße: Anwendungsfälle sind verschieden
 - Meine Images: 50MB für Alles ← das ist viel!
 - Falls GSM als Kanal verwendet wird: <1MB

Crashes im Feld

- Endkunden haben kein besonderes Talent darin, Fehlerfälle gut zu beschreiben...
- Lösung: Schmale Crashdumps an einen Telemetrieserver schicken, der dann passende Debug-Symbole vorliegen hat



Verschiedenes

Security: Das 'S' in IOT steht für Sicherheit

- Alle Updates immer Signieren!
- Alle Updates immer Verschlüsseln!
- Alle Protokolle ohne Crypto ablehnen!
- Selbst wenn User an die Serial-Konsole/UART kommen
 - Passwort (ab dann haben sie es meiner Meinung nach verdient)
- “Burner” SSH Schlüsselpaare auf den Geräten vorhalten
 - Die mit der Zeit abschalten wenn geleakt
 - Beobachtete Opsec: 10 Schlüssel ausdrucken und in den FirmenSafe legen, Dem Notar geben, Dateien danach löschen

Datum und Zeit

- Großes Problem ab dem Punkt wo man mit externen Schnittstellen mit anderer Zeitvorstellung reden muss
 - RTC, Zeitzonen...
- Typisches Problem: Zertifikate
 - Let's Encrypt Certs sind entweder nicht mehr, oder oft *noch nicht* gültig.
- Daumenregel: Sobald man “nach außen” kommuniziert braucht man vernünftige Zeit!

Lizenzen

- Die Menge von bekannter und unbekannter Software erreicht Level in der die Entwickler über Lizenzen keine Aussagen mehr über installierte Software treffen können
 - Es gibt Dedizierte Tools die im eigenen Code nach Lizenzen suchen
- Softwarelizenzen bringen Pflichten mit sich
- Wichtig ist: eine Bill Of Material (BOM) führen
- Für interessierte: Sucht bei modernen Autos die Lizenzliste
→ Ihr werdet sie finden :)

End Of Life

End of Life

- Es gibt Gründe, alle Geräte einzusammeln
- Auch bei Industrieprodukten laufen Zeiten aus
- Es gibt Explizit auf lange Laufzeiten ausgerichtete Hardware /Distributionen, Konferenzen nur darüber (30 Jahre+)
- Der Rest kann ein Sicherheitsrisiko sein
- Killswitch-Update

End Of Vortrag!

Ich beantworte alle Fragen, AMA!