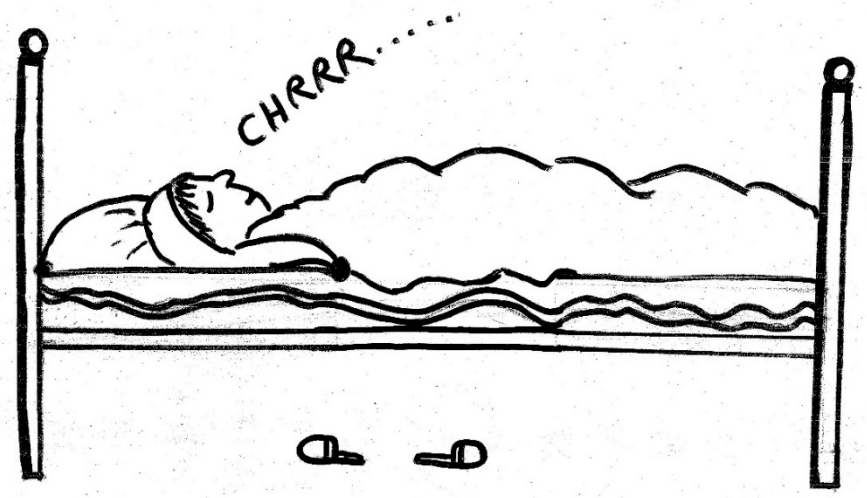
EMBEDDED, RAUS AUS DEM BETT

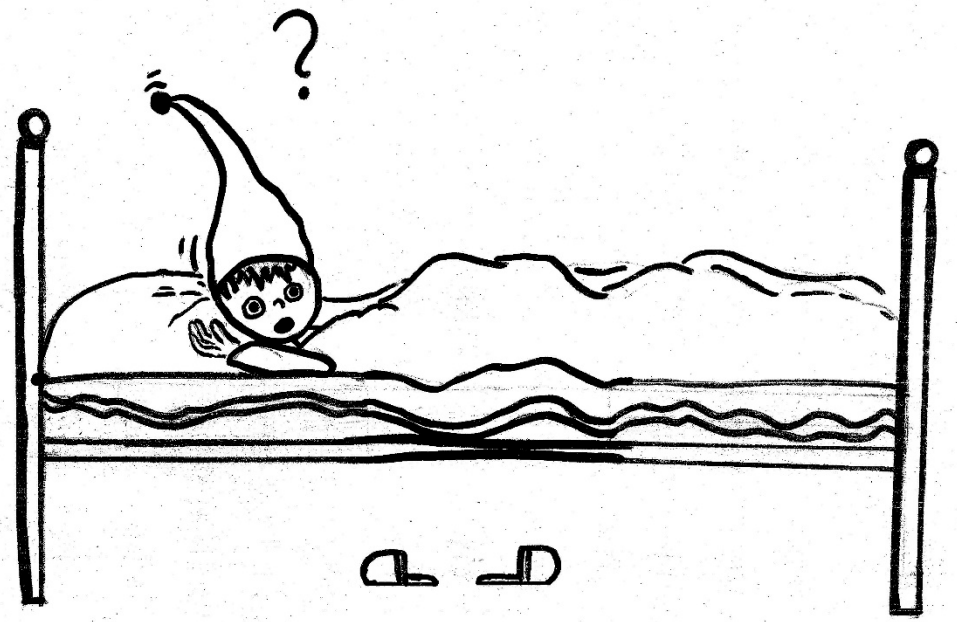


BOSCH

Embedded, raus aus dem Bett
Ich, die Schlafmütze

Über mich

- Harald Göttlicher
- Eule (im Gegensatz zu Lerche)
- Softwarearchitekt bei Robert BOSCH GmbH
- Bereich AA-AS
- Continuous Delivery: seit ~9 Jahren



Embedded, raus aus dem Bett
Ich, die Schlafmütze

Unsere Arbeit, um wach zu bleiben

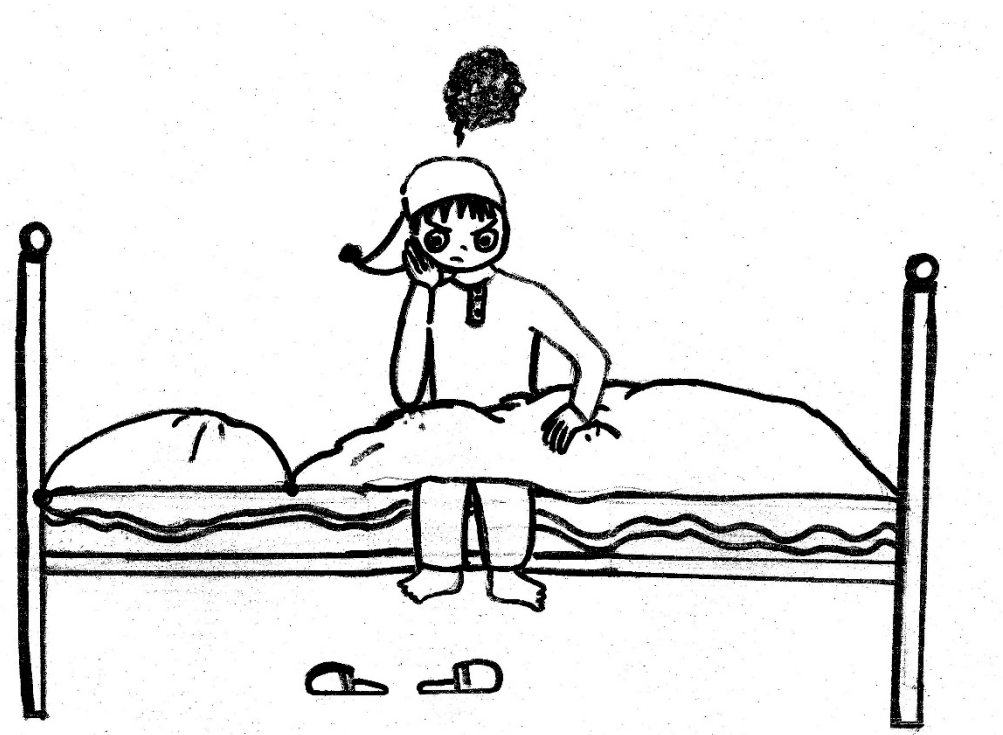
- Ausrüstung für Autowerkstätten
- Hauptthema: Diagnosesoftware www.esitronic.com
- Lange Historie mit vielen Technologien
- „Beide Welten” – Web- & Offline-Produkte
  - WebApps und Services
  - Container
  - VM & Cloud-Plattformen
  - Embedded Linux
  - Embedded Android
  - Desktop Windows

Embedded, raus aus dem Bett
Vorbemerkungen

Bitte nicht über den provokativen Titel ärgern ;-)

- Ich wache auch gerade erst auf
  – wir haben auch noch nicht alles umgesetzt
- Du bist vielleicht schon (teilweise) auf dem hier beschriebenen Weg
- Ich weiß es vielleicht nicht besser
- Wahrscheinlich kenne ich Euren Anwendungsfall nicht gut genug
- Dieser kurze Vortrag kann bei Weitem nicht alles abdecken

Soweit klar – also schauen wir mal genauer hin!

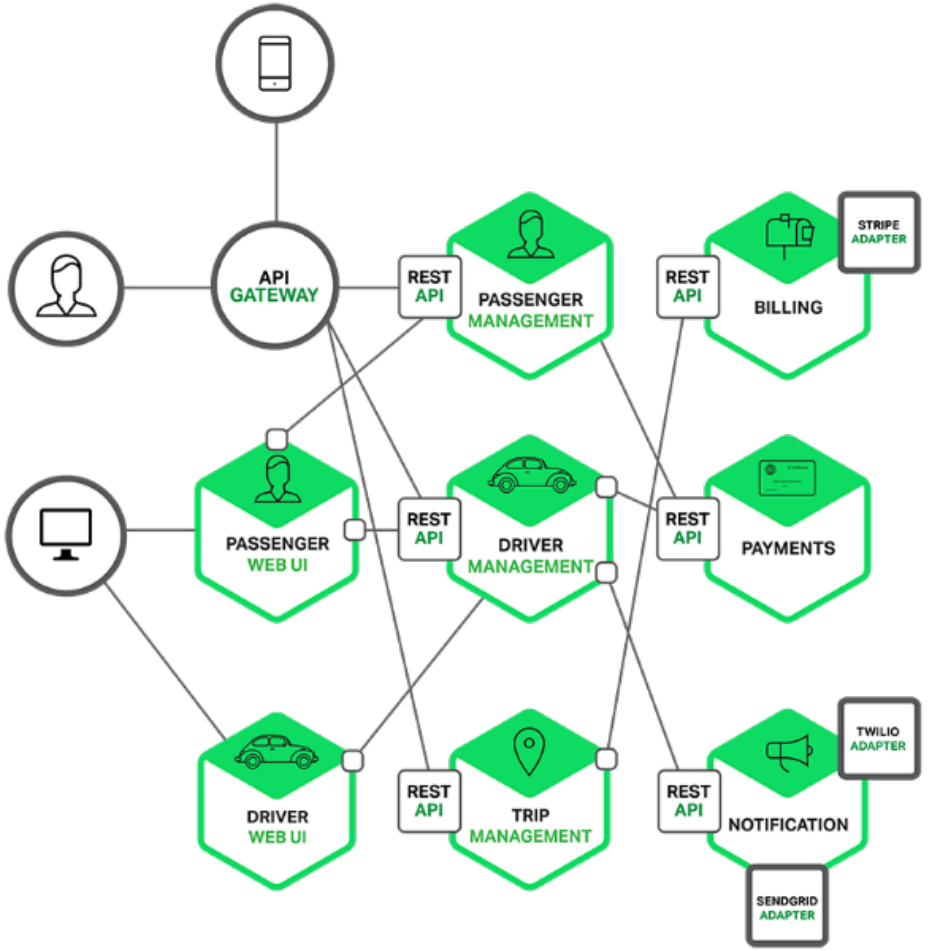


Embedded, raus aus dem Bett
Das Problem

Die Softwarewelt hat sich schnell verändert

- Agile Arbeitsmethoden
- DevOps-Prinzipien
- Microservices
- Containerisierung
- Cloud Computing / IaC
- Automatisierte Tests
- Continuous Delivery / Deployment

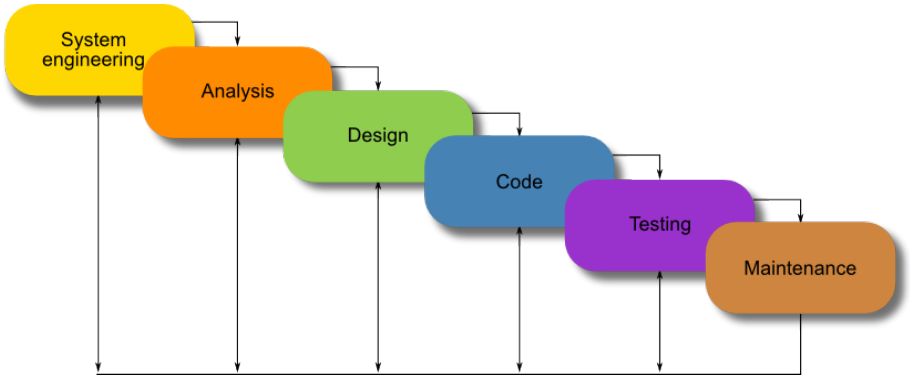
Sind das nur „Buzzwords”?



Embedded, raus aus dem Bett
Das Problem

Embedded- und Desktop-Software haben sich wenig verändert

- Oft immer noch Wasserfall-Denken – selbst wenn agile Teams bestehen!
- Strikte Prozesse und Restriktionen durch Standards
- Kein DevOps-Denken, weil es kein Ops gibt
- Mehr oder weniger Monolithische Software
- Beschränkt auf/gebunden an Hardware
- Teilweise manuelles Testen
- Continuous Integration ist üblich, aber Auslieferung dauert Wochen



Aber ist das notwendigerweise ein Problem?

Embedded, raus aus dem Bett
Das Problem

Ja, ist es: den das alles führt zu viel manueller (kostspieliger) Arbeit in *allen Bereichen*

Manual work	Elite	High	Medium	Low
Configuration Management	5%	10%	30%[a]	30%[a]
Testing	10%	20%	50%	30%
Deployments	5%	10%	30%[b]	30%[b]
Change Approvals	10%	30%	75%	40%

Medians reported because distributions are not normal
[a,b] Not significantly different when testing for differences using Tukey's post hoc analysis

Quelle: "Accelerate: State of DevOps Report"

Embedded, raus aus dem Bett
Das Problem

Und es führt zu

- Niedrigem Durchsatz
- Langsame Lieferung (Time to Market)
- Hohe Fehlerraten
- Langsame Bugfix-Lieferung

Hier sieht man einmal, wie exorbitant die Differenz zu State-of-the-Art-Projekten ist!

[Quelle: “Accelerate: State of DevOps Report”]

Embedded, raus aus dem Bett
Das Problem

Und all das lässt uns viel weniger Zeit für die Entwicklung neuer Features

Time Spent	Elite	High	Medium	Low
NEW WORK	**50%**	**50%**	**40%**	**30%**
Unplanned work and rework	19.5%	20%[a]	20%[a]	20%[a]
Remediating security issues	5%	5%[b]	5%[b]	10%
Working on defects identified by end users	10%	10%[c]	10%[c]	20%
Customer support work	5%	10%	10%	15%

Medians reported because distributions are not normal.
[a] Significantly different when testing for differences using Tukey's post hoc analysis
[b, c] Not significantly different when testing for differences using Tukey's post hoc analysis

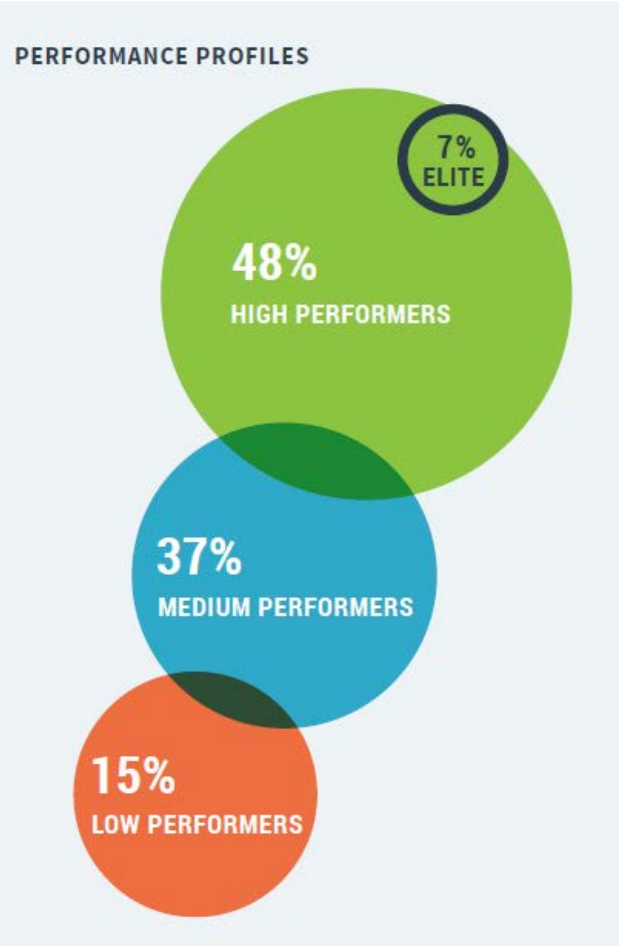
Quelle: "Accelerate: State of DevOps Report"

Embedded, raus aus dem Bett
Das Problem

Also: allgemein – und speziell in diesem Bericht

- Sind die meisten Embedded- und Desktop-Softwareprojekte die zitierten „Low Performers”?
- Sind die meisten „Low Performers” umgekehrt auch Embedded / Desktop?

Der Bericht sagt es nicht – aber ich fürchte:
wahrscheinlich sind sie es!

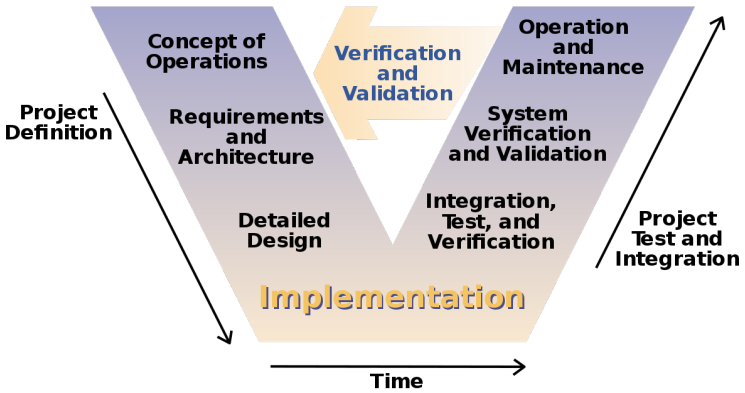


Quelle: “Accelerate: State of DevOps Report”

Embedded, raus aus dem Bett
Das Problem

Aber warum? Einige mögliche Gründe

- Die starren Prozesse großer Firmen
- Traditionelle Marketingstrukturen
- Restriktionen durch Standards wie SPICE, MISRA, AUTOSAR und diverse ISOs
- Komplexe QA-Prozesse
- Legacy-Sourcecode
- Komplexe Tests, die schwierig zu automatisieren sind
- Oft auch „Hardware-in-the-Loop” (HIL) Tests
- Umständliche Auslieferung über Installer oder Flash-Tools

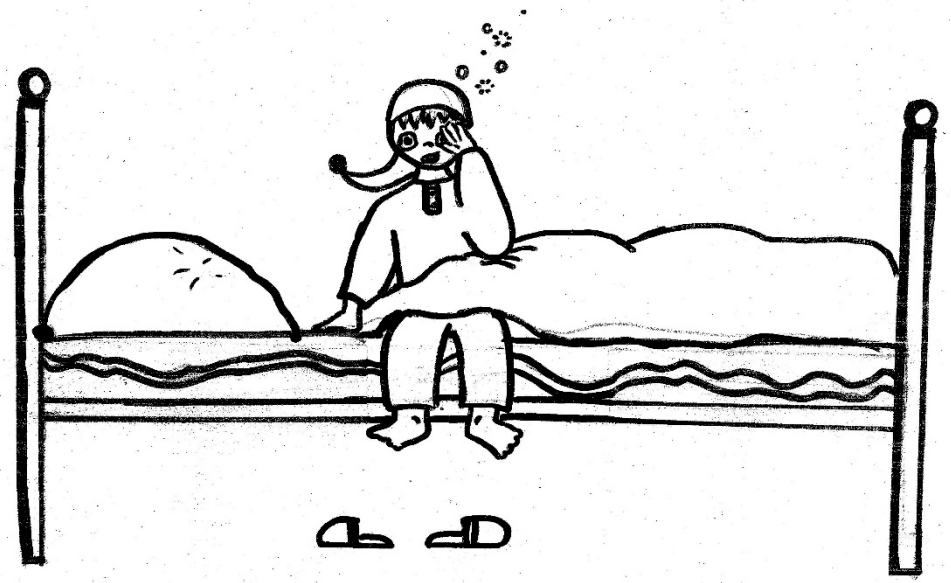


Embedded, raus aus dem Bett
Was nun?

Also, was könnten wir tun?

- Was können wir verändern?
- Was können wir von DevOps lernen oder übernehmen?
- Können wir schnell wie Webprojekte werden?
- Oder stecken wir durch die Einschränkungen fest?

Könnten wir denn „High” oder „Elite Performer“ werden?



Embedded, raus aus dem Bett
Was nun?

Also, passen die etablierten Methoden und Technologien wirklich nicht? Sehen wir es uns einmal an:

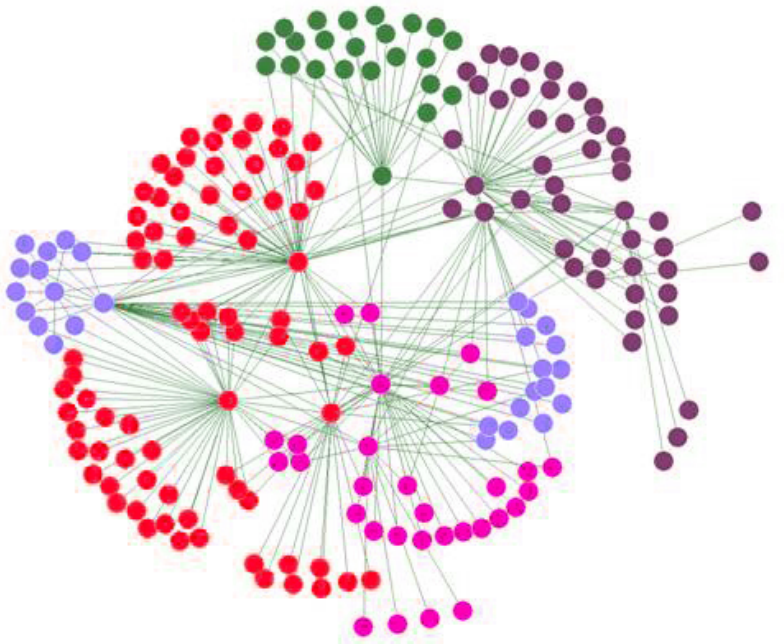
Online-Welt	Embedded- / Desktop-Welt	
Agile Arbeitsmethoden	Wasserfall / Standards und ISOs	☑/☒ ?
DevOps	Kein Ops benötigt	☒ ?
Microservices	Monolithische Software	☑/☒ ?
Containerisierung	Beschränkt auf / gebunden an Hardware	☒ ?
Cloud-Computing		
Automatisierte Tests	Teilweise manuelle Tests / HIL	☑/☒ ?
Continuous Delivery / Deployment	CI + Installer / Flashing / Manuell	☑/☒ ?

→ **Wirklich?** Oder könnten wir *etwas ähnliches* tun, so dass wenigstens *die Ideen* passen?

Embedded, raus aus dem Bett
Was nun?

Meine Vision ist…

- Die Monolithen in wirklich sehr, sehr viele einzelne Komponenten aufzuteilen
- Diese Komponenten schnell und unabhängig voneinander zu bauen, zu testen und freizugeben
- Updates beim Anwender auf einzelne Komponenten durchzuführen, anstatt großer Pakete



Aber auch

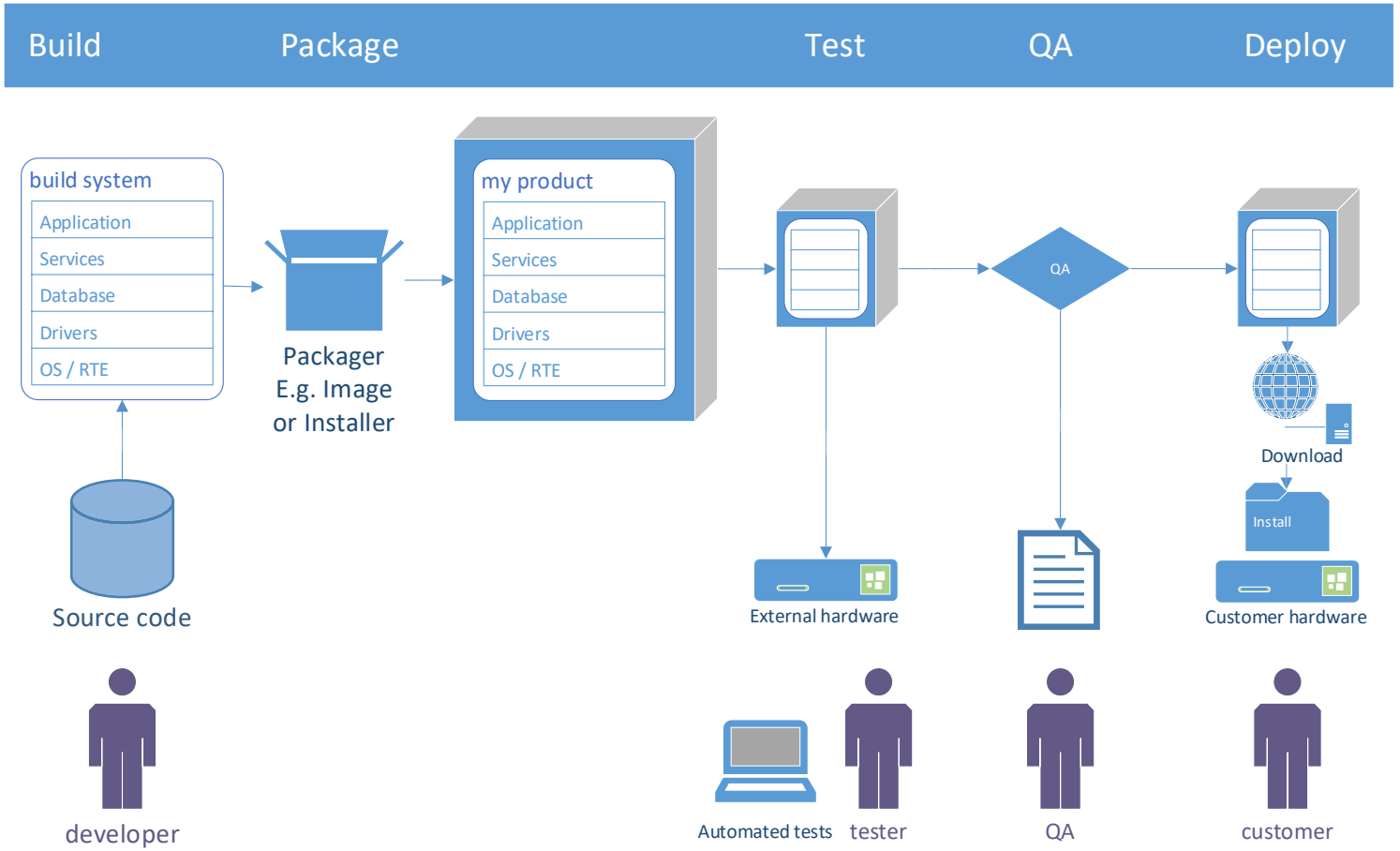
- Schnelle, schlanke Prozesse, die das erst ermöglichen
- Kulturelle Änderungen ähnlich DevOps, um es zu unterstützen

Wie? Gehen wir tiefer in die Details.

Embedded, raus aus dem Bett
Der große Schnitt

Schaut Euren Monolithen an

- Viele Produkte bestehen aus einem oder wenigen Teilen
- Die Aufteilung wird oft nur nach Layern gemacht
- Tests haben oft einen großen Umfang
- QA wird manuell mit Dokumenten und Meetings durchgeführt
- Packaging & Deployment ist ebenfalls groß & monolithisch



Embedded, raus aus dem Bett
Der große Schnitt

Schauen wir doch mal in die Online-Welt - Microservices

Embedded, raus aus dem Bett
Der große Schnitt

Schauen wir doch mal in die Online-Welt - Microservices

> *Microservices are a software development technique(...) that structures an application as a collection of loosely coupled services. In a microservices architecture,* ***services are fine-grained*** *and the protocols are* ***lightweight****. The* ***benefit of decomposing*** *an application into different smaller services is that it improves modularity. This makes the application* ***easier to understand, develop, test****, and become more* ***resilient*** *to architecture erosion. It* ***parallelizes development*** *by* ***enabling small autonomous teams*** *to develop, deploy and scale their respective services independently. It also allows the* ***architecture*** *of an individual service to* ***emerge through continuous refactoring****. Microservices-based architectures* ***enable continuous delivery and deployment****.*
>
> *Wikipedia on Mar 24, 2019*

- Sind das nicht die Eigenschaften, die wir uns schon immer wünschen?
- Könnten wir manche davon für Embedded- und Desktopsoftware übernehmen?

Embedded, raus aus dem Bett
Der große Schnitt

Zunächst brauchen wir einen Paradigmenwechsel von einer Schichten-Architektur zu eigenständigen Komponente für jeweils eine Funktion.

Also, keine Angst: schneidet Euren Monolithen!

- Schneide, teile, zerlege – kleiner als Du jemals gedacht hast!
- Keine Angst vor tausenden oder zehntausenden Komponenten – das ist handhabbar!
- Beginne zunächst langsam an natürlichen Komponentengrenzen
- Aber behalte die „Microservice”-Idee im Auge: eine Komponente pro Funktion und sauber definierte Interfaces
- Lasse auch zunehmend unabhängige (Weiter-)Entwicklung der Komponenten zu

Embedded, raus aus dem Bett
Der große Schnitt

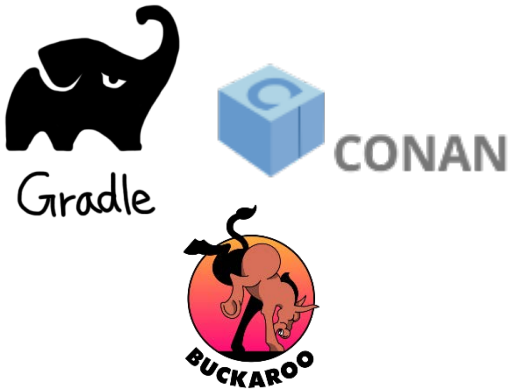
Wie handhaben wir diesen Flohzirkus?

- **Automatisierung der Automatisierung** ist das Schlüsselwort!
- Setze Deine Continuous Delivery (CD) automatisiert auf
- Verwende Infrastructure-as-Code (IaC)- und Change-Management (CM)-Tools
  - Z.B. Container-Cluster wie OpenShift, Docker Enterprise
  - Z.B. Ansible, Chef, Puppet
- Setze Jenkins-Master in Containern und mit JCasC auf
- Repliziere und teste Deine gesamte CD-Landschaft für Dev und QA
  - Wir reden hier von automatisierten Tests der CI/CD selbst!

Embedded, raus aus dem Bett
Der große Schnitt

Welche Technologien kann ich verwenden?

- Eine modulare/Template-basierte CI/CD kann riesige Mengen von Jobs verwalten und skalieren
- Dazu gibt es z.B. folgende Jenkins-Technologien
  - Pipeline-Libraries für mehrere ähnliche Build-Aktionen
  - Multibranch, JobDSL oder Templates für ähnliche Build-Jobs
- Bestehende Package-Manager verwenden
  - Z.B. für C++: Conan, Gradle, Buckaroo
- Container als Build-Agents
  - Skalierende Lösung, die Tools und OS-Versionen kapselt

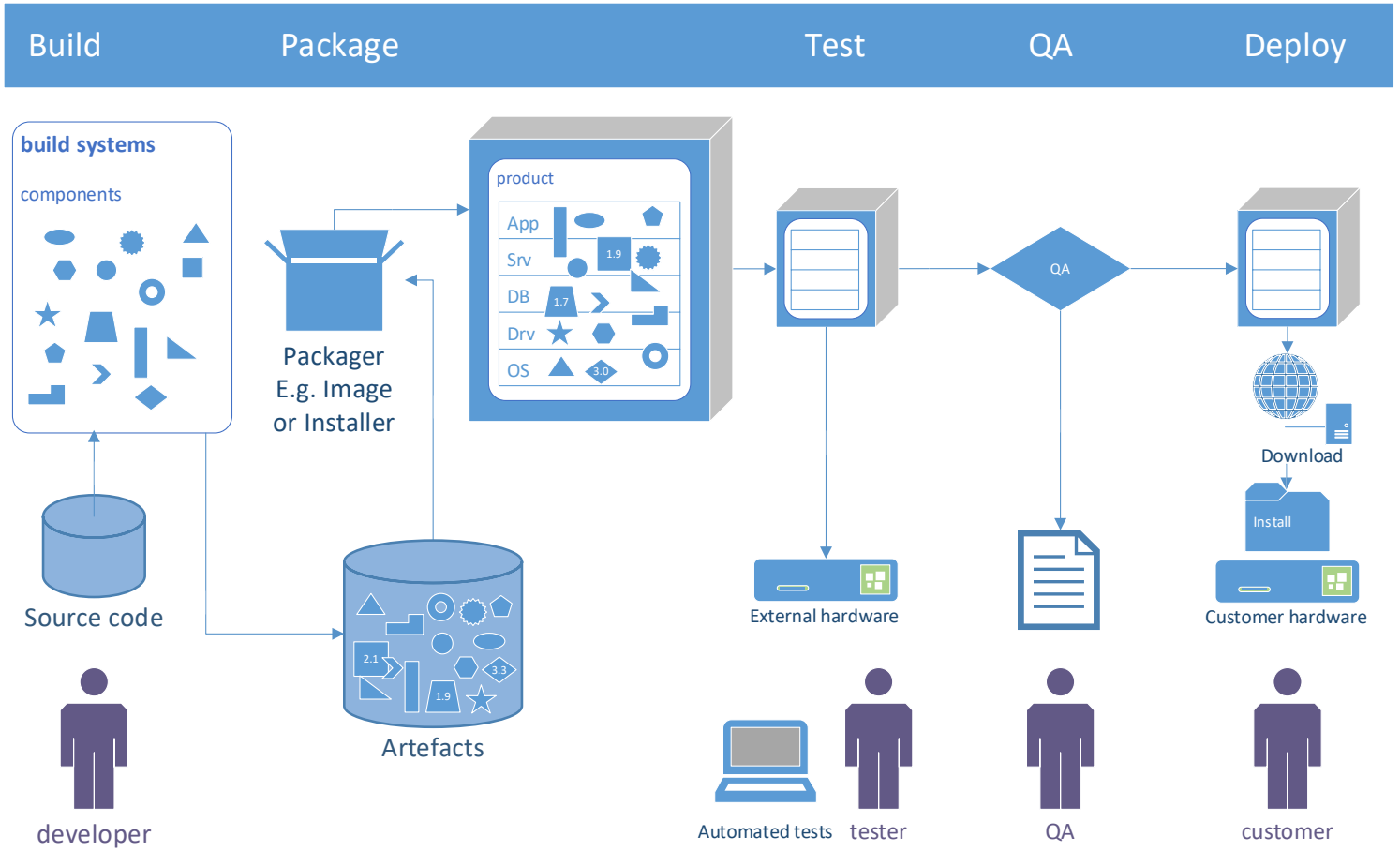


Damit bist Du auch für zehntausende Komponenten gerüstet!

Embedded, raus aus dem Bett
Der große Schnitt

So könnte es nach dem ersten Schritt aussehen

- Entwicklung und Build konzentrieren sich nun auf viele kleine Komponenten
- Komponenten sind bereits als Artefakte gespeichert
- Packaging, Test, Deployment noch unverändert
- Prozesse müssen also zunächst nicht angepasst werden



Embedded, raus aus dem Bett
Atomare Updates

Nun brauchen wir einen Paradigmenwechsel von Installern zum Update einzelner Komponenten oder Dateien.

Die Idee ist

- Einen atomaren Updatemechanismus zu entwickeln
- Einzelne Komponenten zu aktualisieren
- Nur die geänderten Teile zu aktualisieren
- Updates sehr häufig durchzuführen

Damit können auch große Installationen sehr schnell aktualisiert werden.

Dies ist für jegliche Dateibasierte Systeme möglich.

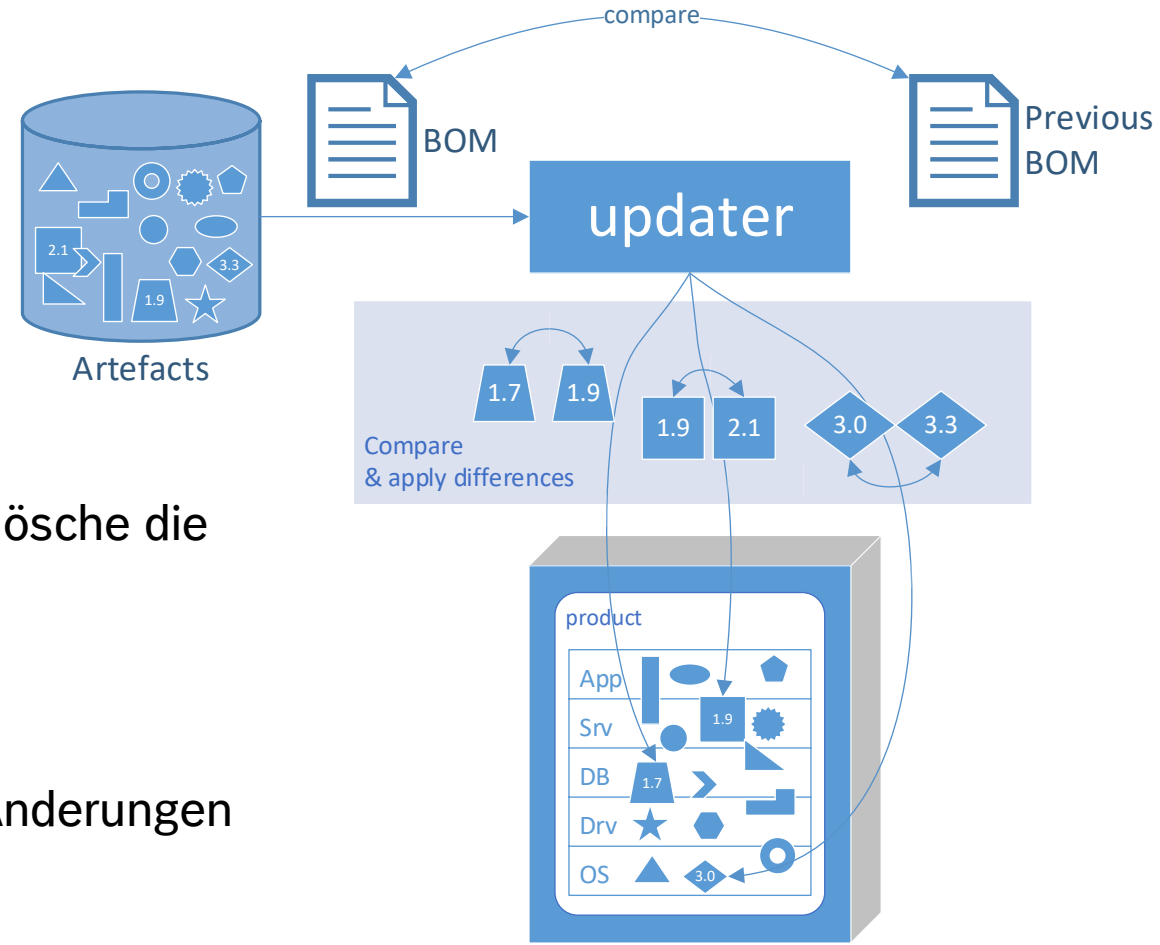
Ich sehe dies als einen der entscheidendsten Teile an.

Embedded, raus aus dem Bett
Atomare Updates

Das ist das “Rezept” für den atomaren Updater

- Erstelle ein BOM (bill of materials)-Snapshot Deines ganzen Produkts
- Vergleiche mit dem letzten BOM-Snapshot
- Lade aktuelle und vorhergehende Abhängigkeiten bzw. Artefakte herunter
- Kopiere die neuen oder geänderten Dateien, lösche die entfernten Dateien*
- Speichere bei Erfolg die letzte BOM
- Wiederhole für jedes Update

Führt man diese regelmäßig durch bleiben die Änderungen klein und die Updates sind sehr schnell.



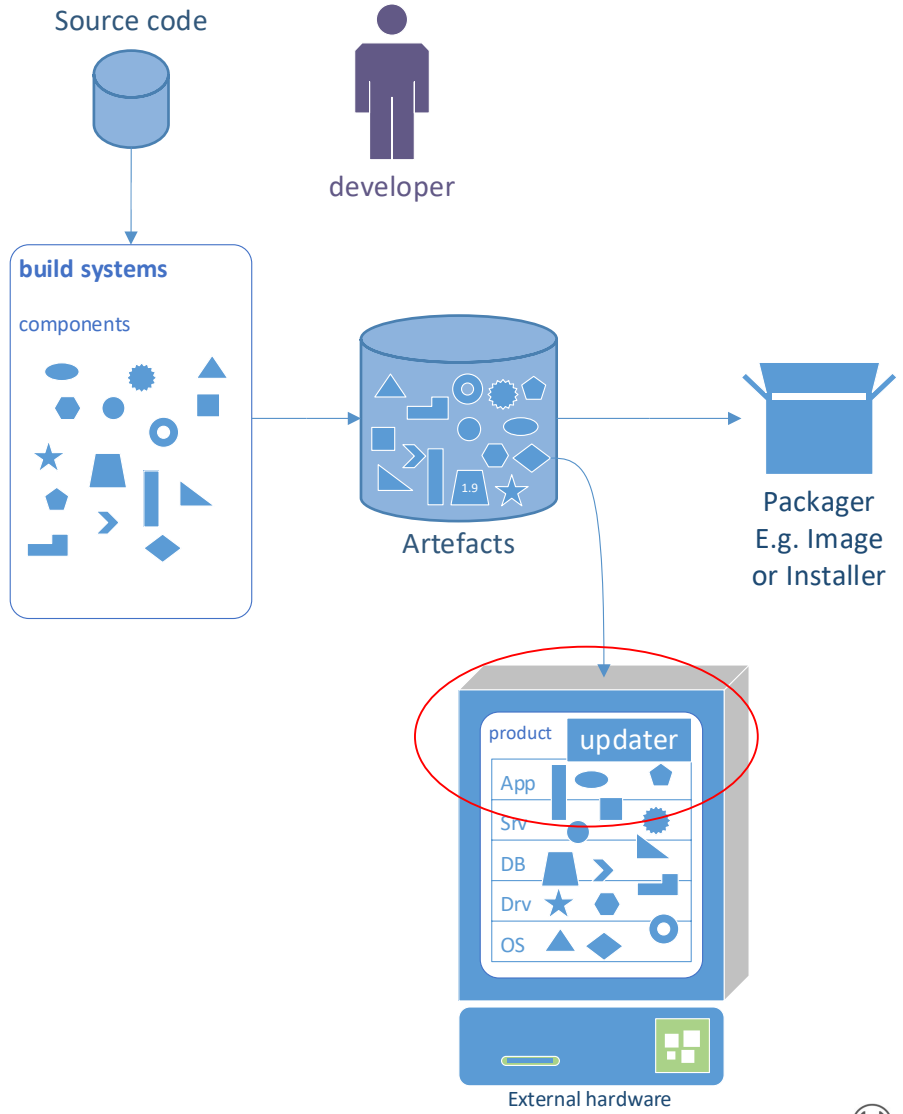
* Annahme: eine Komponente enthält mehrere Dateien

Embedded, raus aus dem Bett
Atomare Updates

Integriere den Updater

- Updater als Teil des Produkts
- Aktiv aktualisieren mit „Pull“-Updates statt „Push“

Das ist bereits eine Vorstufe der späteren produktiven Nutzung.



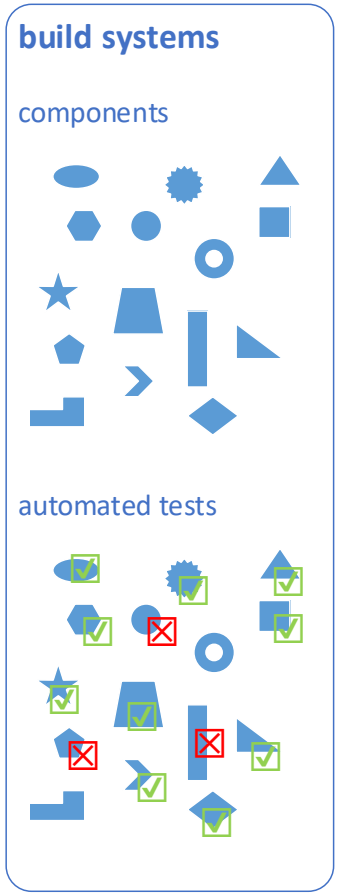
Embedded, raus aus dem Bett
Atomares Testen

Auch die Tests müssen wie die Komponenten geschnitten werden

- Eine automatisierte Test-Suite pro Komponente
- Unabhängig und schnell lauffähig

Das richtige Triggern der Tests

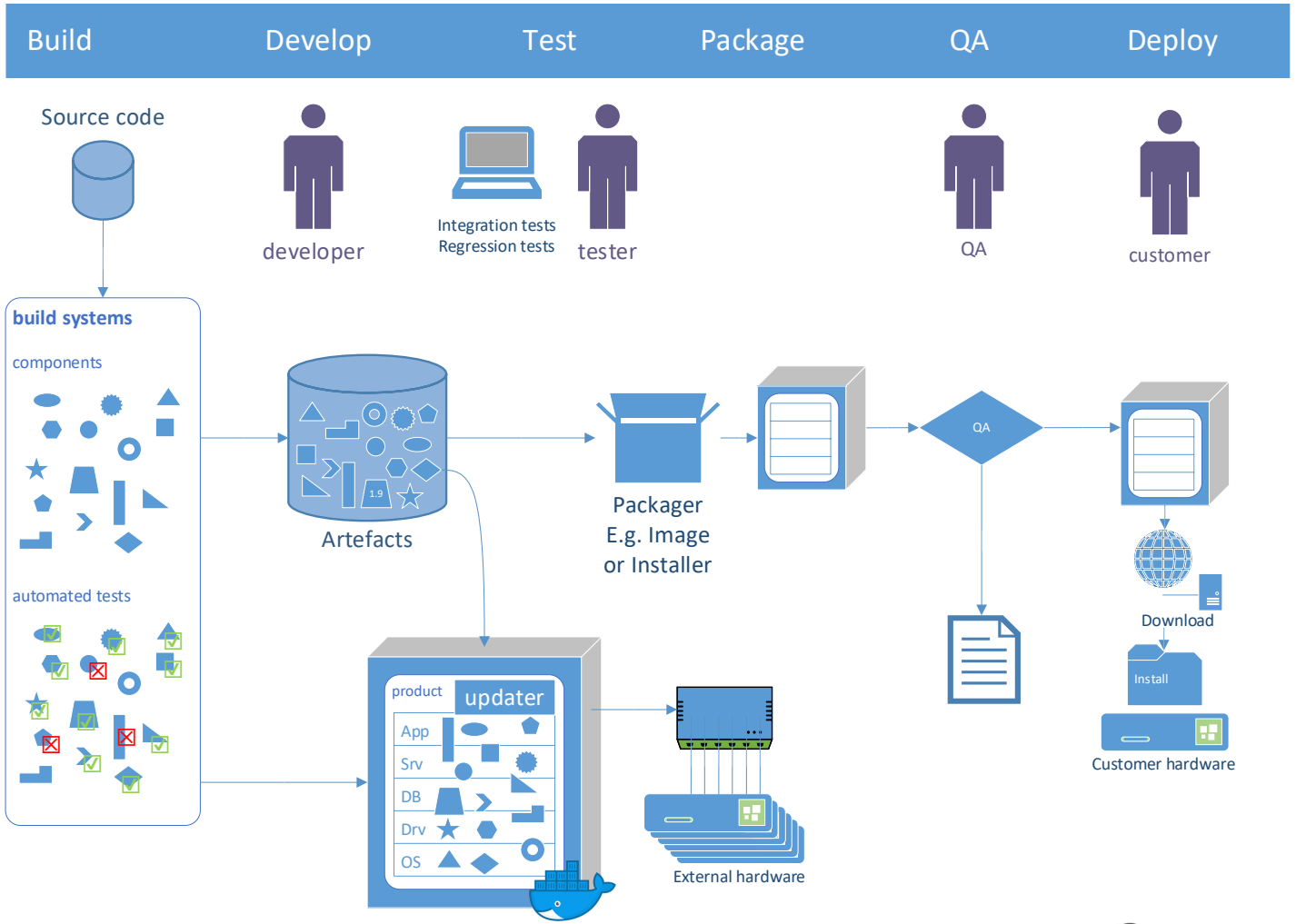
- Die CI/CD hat das genaue Wissen, was geändert wurde
- Triggere komponentenweise Tests direkt nach Build und atomarem Update
- Starte nur Tests der geänderten Komponenten



Embedded, raus aus dem Bett
Atomares Testen

Container verwenden

- Starte Tests in Containern
- Ein Container pro Schicht oder noch feingranularer
- Stelle auf HAL tests in Containern um
- Z.B. mit einem USB-Ethernet-Hub
  - Es gibt auch Hubs mit verschlüsselter Kommunikation (suche nach “dongle server”)



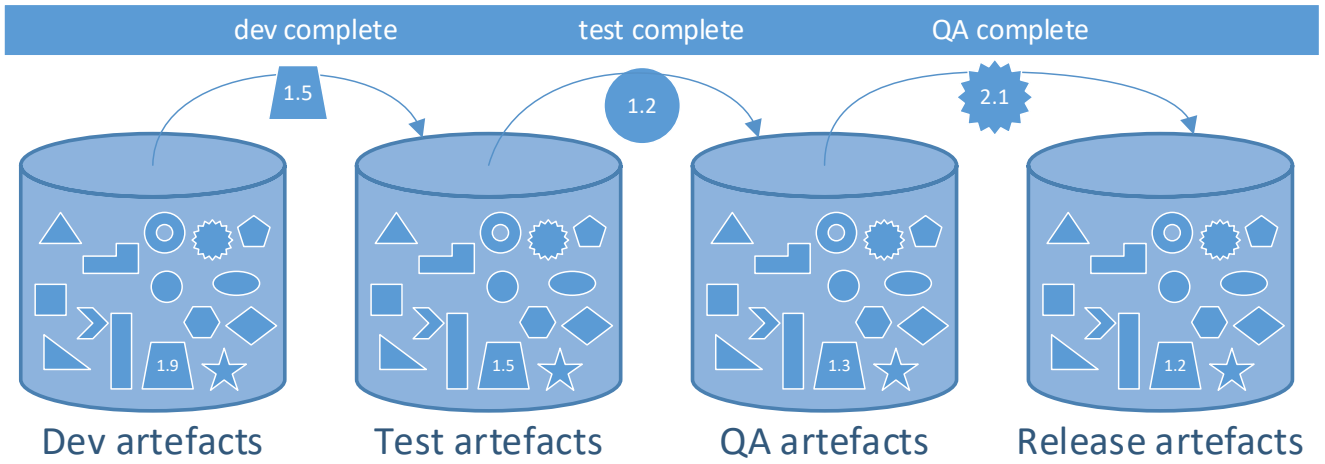
Embedded, raus aus dem Bett
Atomarer Releaseprozess

Einen automatisierten Releaseprozess etablieren

- Test/QA nur als propagieren von Artefakten, beispielsweise
  - Fertig entwickelte Artefakte werden auf der Stufe “Test” veröffentlicht
  - Getestete Artefakte werden auf die Stufe “QA” veröffentlicht
  - Nach QA/Acceptance Test, Veröffentlichung auf Stufe “Release”

Wenn manuelle Freigabe nötig ist

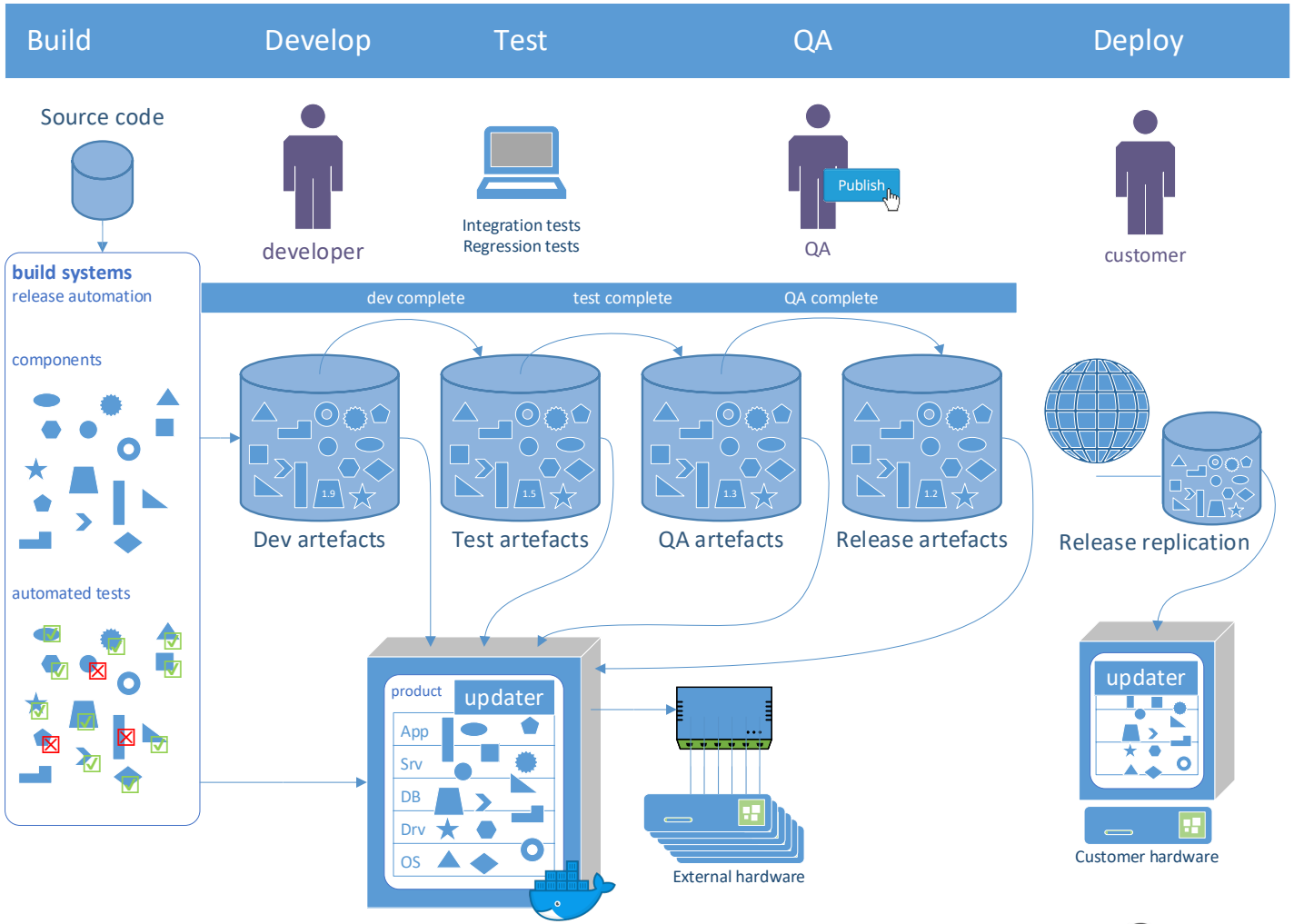
- Einfach auf Knopfdruck!



Embedded, raus aus dem Bett
Atomare Auslieferung

„Releaseloses" automatisiertes Update im Produktivbetrieb

- Atomares Update vom Testen wird hier wiederverwendet
- Robust gemachter Updater wird Teil des Produkts
- Aktive Updates mit „Pull"-Mechanismus statt „Push"
- Kleine atomare Updates liefern
- Keine Installer oder Release
- Schnelle Auslieferung – wie in der Online-Welt!



Embedded, raus aus dem Bett
Atomare Auslieferung

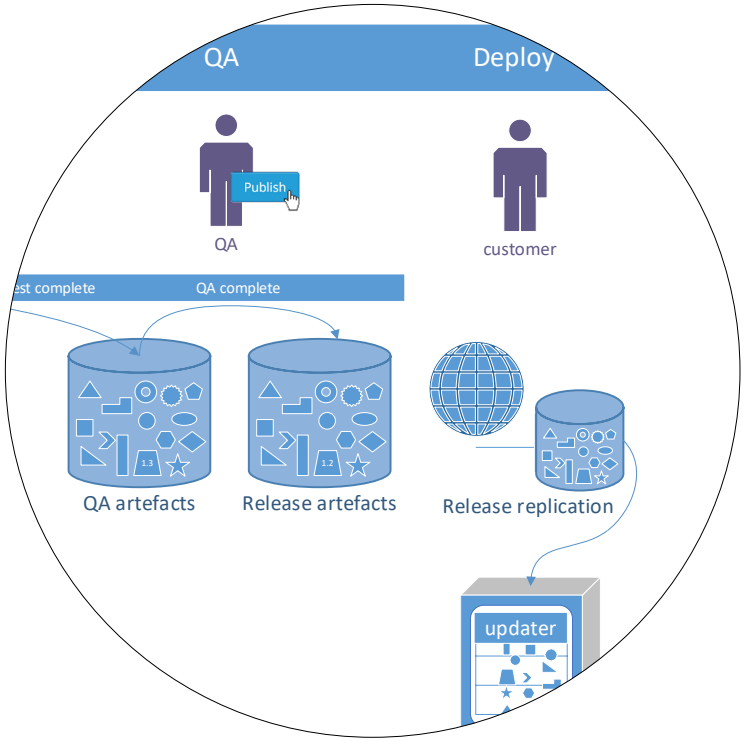
Ermöglicht uns das „Releaselose” Szenario

- Wöchentlich zu liefern?
- Täglich?
- Innerhalb von Stunden?
- Noch öfter?

Nein!

Es ist viel besser

- Wir liefern *was fertig ist*
- Innerhalb der gleichen Minute!



Oder anders gesagt: selbst *wenn* es Tage dauert, wird es dann *sofort* geliefert.

Embedded, raus aus dem Bett
Prozessänderungen

Einführung agiler Prozesse

- Agile Methoden verwenden oder von ihnen lernen
- Evtl. manches von bereits agil arbeitenden anderen Teams oder Abteilungen übernehmen

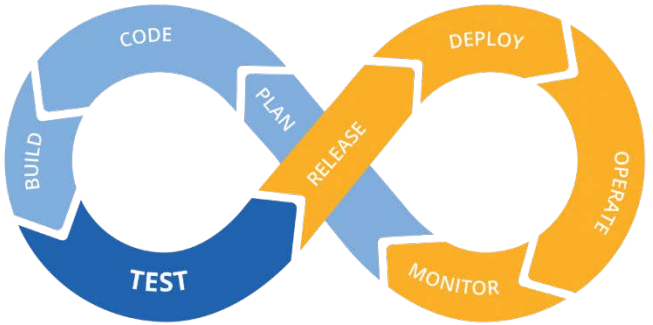
Schnelle Freigabe & QA ist entscheidend

- Umstellung auf schlanke, schnelle QA- und Freigabeprozesse
  - Auch hier: lernen von bereits agil arbeitenden Teams der Firma
- Prüfen, ob und wo manuelle Schritte wirklich nötig sind
- Automatisieren, wo immer möglich
- Manuelle Freigaben nur auf Knopfdruck
  - Meetings und Dokumente vermeiden

Embedded, raus aus dem Bett
Kulturelle Änderungen

Ideen der DevOps- / BizDevOps-Kultur übernehmen

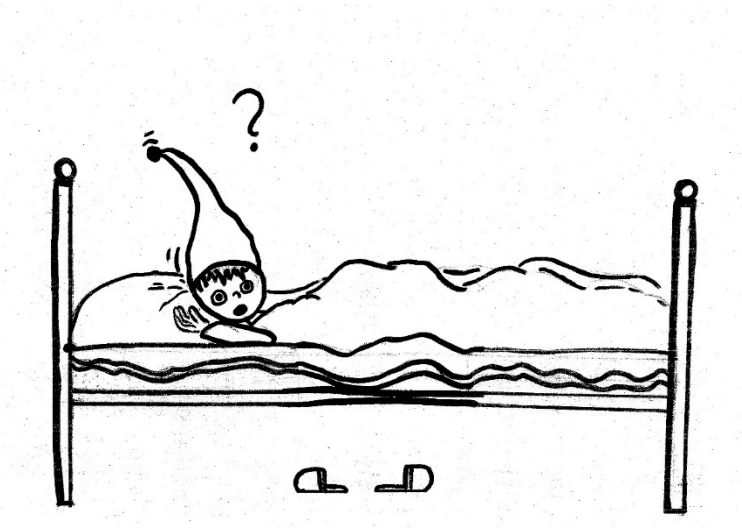
- Effiziente Tools, Infrastruktur und Prozesse nahtlos aufeinander abstimmen
- Auch Metriken und Berichte auf DevOps-Ideen umstellen
- Schnelle, effiziente Kooperation der Teams in Projektmanagement, Entwicklung, Test und QA
  - Idealerweise auch Produktmanagement, Marketing und Management einbeziehen (BizDevOps)
  - Dort ist Ops dann üblicherweise weniger das Thema
- Restrukturieren, um kleine, komponenten-zentrierte Teams zu ermöglichen und zu stärken
- Ende-zu-Ende-Verantwortlichkeit für Komponenten fördern (Sourcecode bis Produktion)



Embedded, raus aus dem Bett
Wie geht es weiter?

Starte in Deiner Organisation

- Fange an, Monolithen zu schneiden
  - Verwende existierende Dependency-Manager
- Starte den Paradigmenwechsel zu atomaren service-artigen Komponenten
- Lerne, wie man sehr, sehr viele CI/CD-Pipelines beherrscht
- Setze komponenten-weise Builds, Updates und Tests auf
- Probiere IaC & Config-Management aus
  - Mit Containerplattformen und Tools wie Ansible, Chef, Puppet
  - Automatisiere die Automatisierung!
- Teste die CI/CD automatisiert!
- Probiere Container für Build & Tests aus
- Passe Prozesse an, wo dies leicht möglich ist



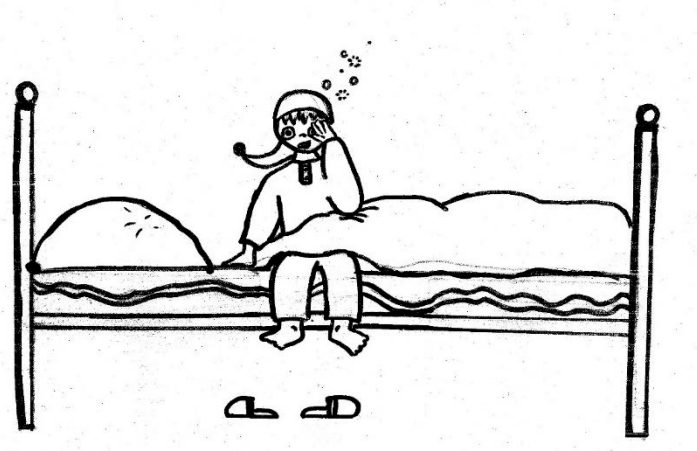
Embedded, raus aus dem Bett
Wie geht es weiter?

Vision

- Starte die Diskussion über neue Arten des Updates in der Produktivumgebung

Und draußen in der Welt

- Versuche, die Gremienarbeit Deiner Firma zu beeinflussen, um Standards anzupassen
  - Es kann ein langer Weg werden – also warum nicht gleich starten?
- Entwickeln wir gemeinsam ein atomares OSS Updater-Tool?
- Schreiben wir ein Buch “Embedded, raus aus dem Bett”?
- …Eure Ideen!



Embedded, raus aus dem Bett
Zusammenfassung

Also, passen die etablierten Methoden und Technologien wirklich nicht? Sehen wir noch einmal hin:

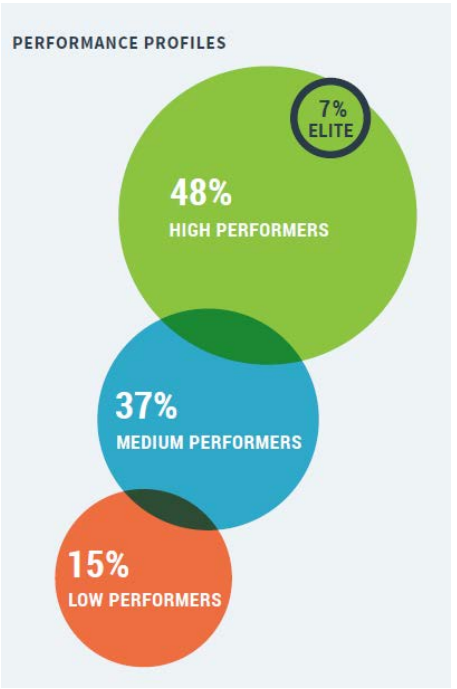
Online-Welt	Embedded- / Desktop-Welt	
Agile Arbeitsmethoden	Agile Methoden + angepasste Standards	☑
DevOps	Übernehmen / Lernen von DevOps	☑
Microservices	Kleine Komponenten / Services	☑
Containerisierung	Container für Build und Test verwenden	☑
Cloud-Computing		
Automatisierte Tests	Atomare HIL-Tests (in Containern)	☑
Continuous Delivery / Deployment	Atomare Updates ermöglichen CD	☑

Zu optimistisch? Versuchen wir, es hinzubekommen und wir werden sehen…

Embedded, raus aus dem Bett
Zusammenfassung

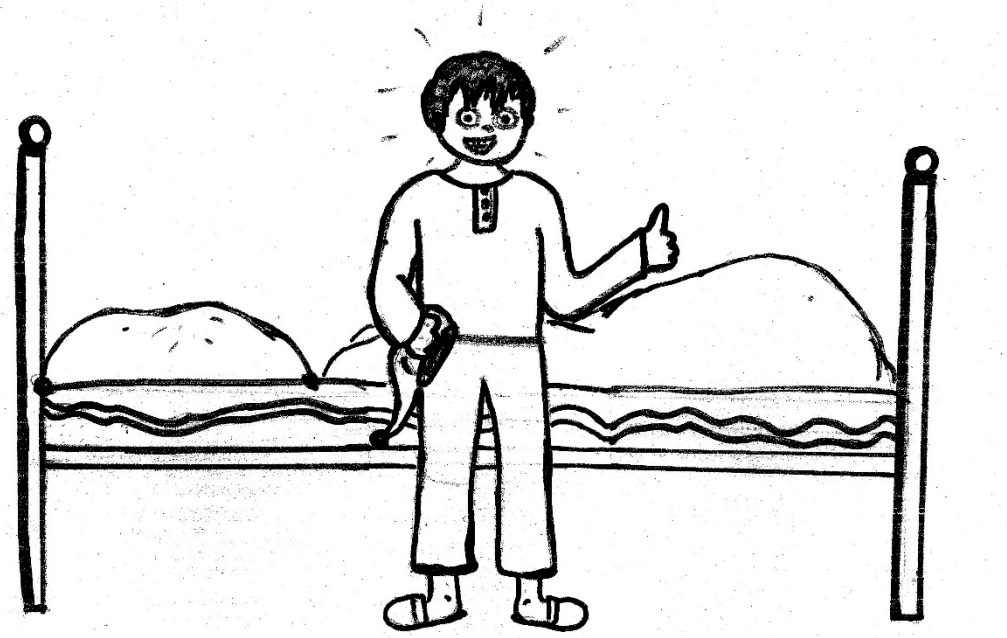
Und denkt daran, warum wir das alles gerne hätten

- Weniger (kostspielige) manuelle Arbeit *in allen Bereichen*
- Höherer Durchsatz
- Schnellere Lieferung (Time to Market)
- Niedrigere Fehlerraten
- Schnellere Bugfix-Lieferung
- Mehr Zeit für neue Features



Danke fürs Wachbleiben!
Zeit für Fragen…

Kontakt:
harald.goettlicher@de.bosch.com
harryg@harryg.de



Schlafmützen-Zeichnungen © Mara Gwendolin (12)