# Von „Accidental Quality“ zu „Quality by Design“

INNOVATE. INTEGRATE. EXCEED.

inovex

inovex

# Hallo,

## ich bin Ulrich Becker

Software Architect bei inovex

- Schwerpunkt: Medical IoT
- Trainings, Architekturanalyse, Dokumentation
- Aktives iSAQB Mitglied

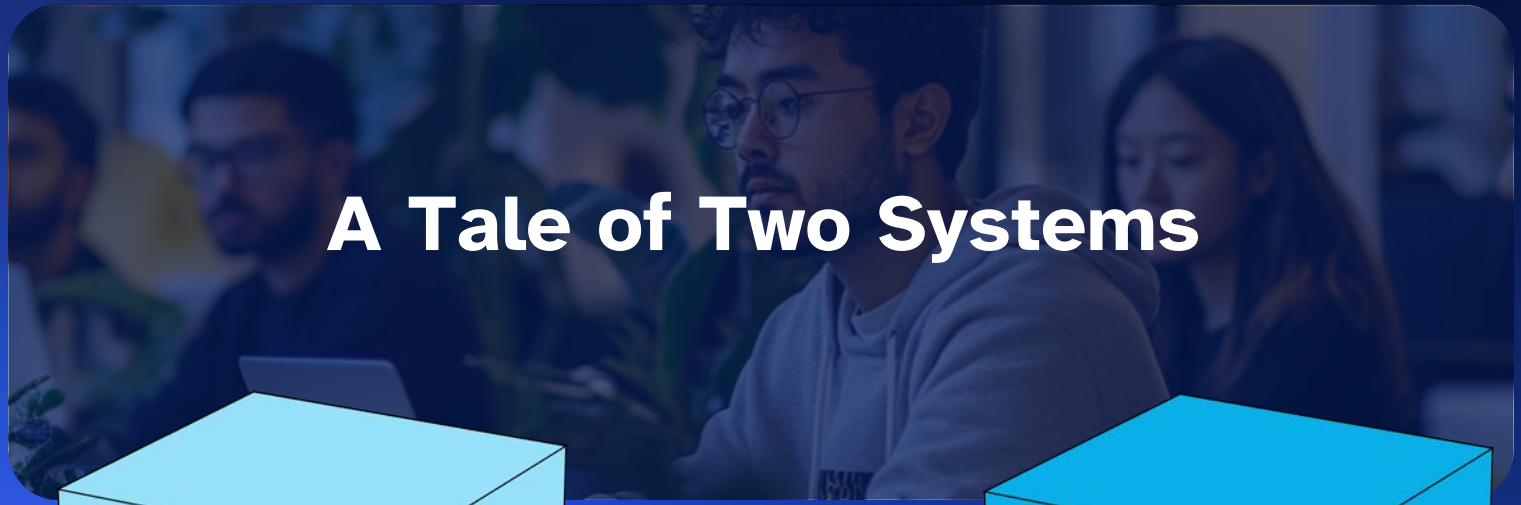

ulrich.becker@inovex.de

+49 174 / 7065171

ulrichbecker

inovex

# A Tale of Two Systems




inovex

# A Tale of Two Systems


Ein **Medizingerät** für **Point-of-Care Tests**, das auf verschiedene respiratorische Erkrankungen mit **hoher Empfindlichkeit** und **Spezifität** testet.
(Influenza A/B, Covid19, RSV, Rhinoviren)


Ein **Medizingerät** für **Point-of-Care Tests**, das auf verschiedene respiratorische Erkrankungen mit **hoher Empfindlichkeit** und **Spezifität** testet.
(Influenza A/B, Covid19, RSV, Rhinoviren)

inovex

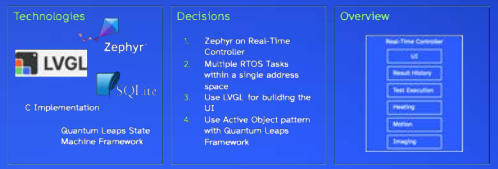
→ **Zwei Systeme mit identischer Kernfunktionalität**

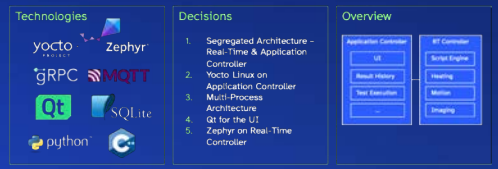
**...aber sehr unterschiedlicher technischer Lösung...**

inovex

# Architektur: InstaDx LC vs. InstaDx Pro





inovex

# Architektur: InstaDx LC vs. InstaDx Pro


## Decisions

1. Zephyr on Real-Time Controller
2. Multiple RTOS Tasks within a single address space
3. Use LVGL for building the UI
4. Use Active Object pattern with Quantum Leaps Framework


## Decisions

1. Segregated Architecture – Real-Time & Application Controller
2. Yocto Linux on Application Controller
3. Multi-Process Architecture
4. Qt for the UI
5. Zephyr on Real-Time Controller

inovex

# Architektur: InstaDx LC vs. InstaDx Pro


Overview

Real-Time Controller

- UI
- Result History
- Test Execution
- Heating
- Motion
- Imaging


Overview

Application Controller

- UI
- Result History
- Test Execution
- ...

RT Controller

- Script Engine
- Heating
- Motion
- Imaging

inovex

# Architektur: InstaDx LC vs. InstaDx Pro


Technologies

Zephyr

LVGL

SQLite

C Implementation

Quantum Leaps State Machine Framework


Technologies

yocto PROJECT

Zephyr

gRPC

MQTT

Qt

SQLite

python

inovex

**Welches System hat die bessere Architektur?**

**It depends...**

**Anforderungen!**

inovex

**Die Anforderungen zur Kern-Funktionalität sind es offensichtlich nicht!**

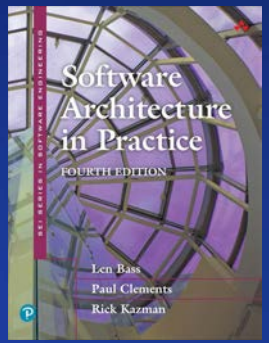
inovex

# Funktionalität ist nicht der primäre Treiber für Architektur!

[...] functionality does not determine architecture.

That is, given a set of required functionality, there is no end to the architectures you could create to satisfy that functionality

[...] if functionality were the only thing that mattered, you wouldn't have to divide the system into pieces at all: A single monolithic blob with no internal structure would do just fine.



inovex

# Unterschiedliche Qualitätsziele und Randbedingungen!


## Mission Statement

InstaDx LC is an affordable device for **fast**, **precise**, and **cost efficient** medical point of care tests.


## Mission Statement

InstaDx Pro is a flexible **connected platform** for medical point of care tests that are **fast**, **precise**, and **cost efficient**.

inovex

# Unterschiedliche Qualitätsziele und Randbedingungen!


Quality Goals & Constraints

- **Portable** to different hardware
- Runs on very **resource constrained** hardware


Quality Goals & Constraints

- **Portable** to different hardware
- Easy to **integrate** with external services
- Easy to **extend** for new tests

inovex

## Unterschiedliche Qualitätsziele und Randbedingungen!



€ Durch den geringen Preis werden viele Geräte verkauft

€ Flexibilität ermöglicht Wachstum der Plattform, mehr Umsatz pro Gerät

inovex

## Was ist eigentlich „Qualität“?

degree to which a software product satisfies stated and implied needs when used under specified conditions

(ISO 25010:2011)

inovex

SWEBOK

# Qualitätsmerkmale

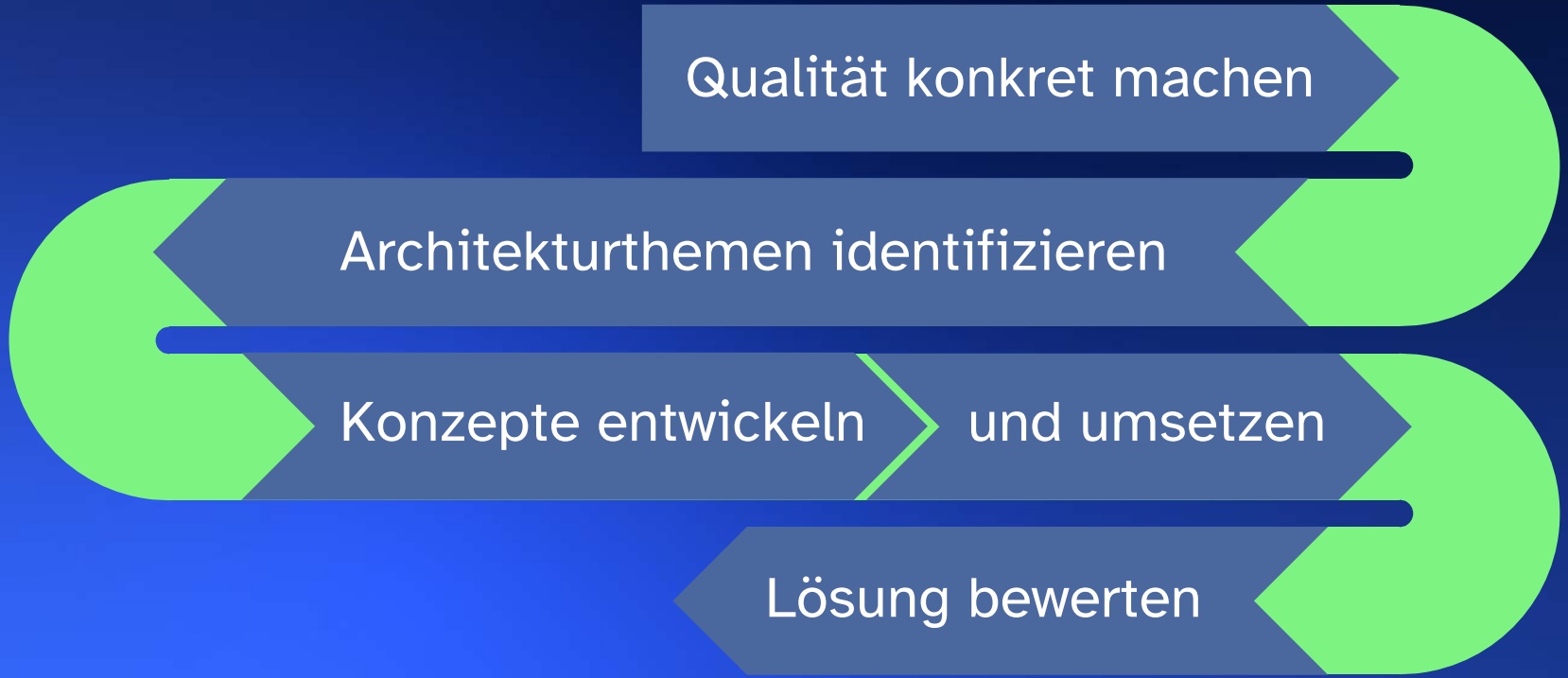
Various attributes contribute to the quality of a software design, including various "**ilities**" (**modularity**, **maintainability**, **portability**, **testability**, **usability**) and "**nesses**" (**correctness**, **robustness**). Qualities are a major subset of concerns [...]

Some qualities can be **observed at runtime** (e.g., performance, security, availability, functionality, usability); **others cannot** (e.g., modifiability, portability, reusability, testability); [...]

*Software Engineering Body of Knowledge zu “Quality Attributes”*

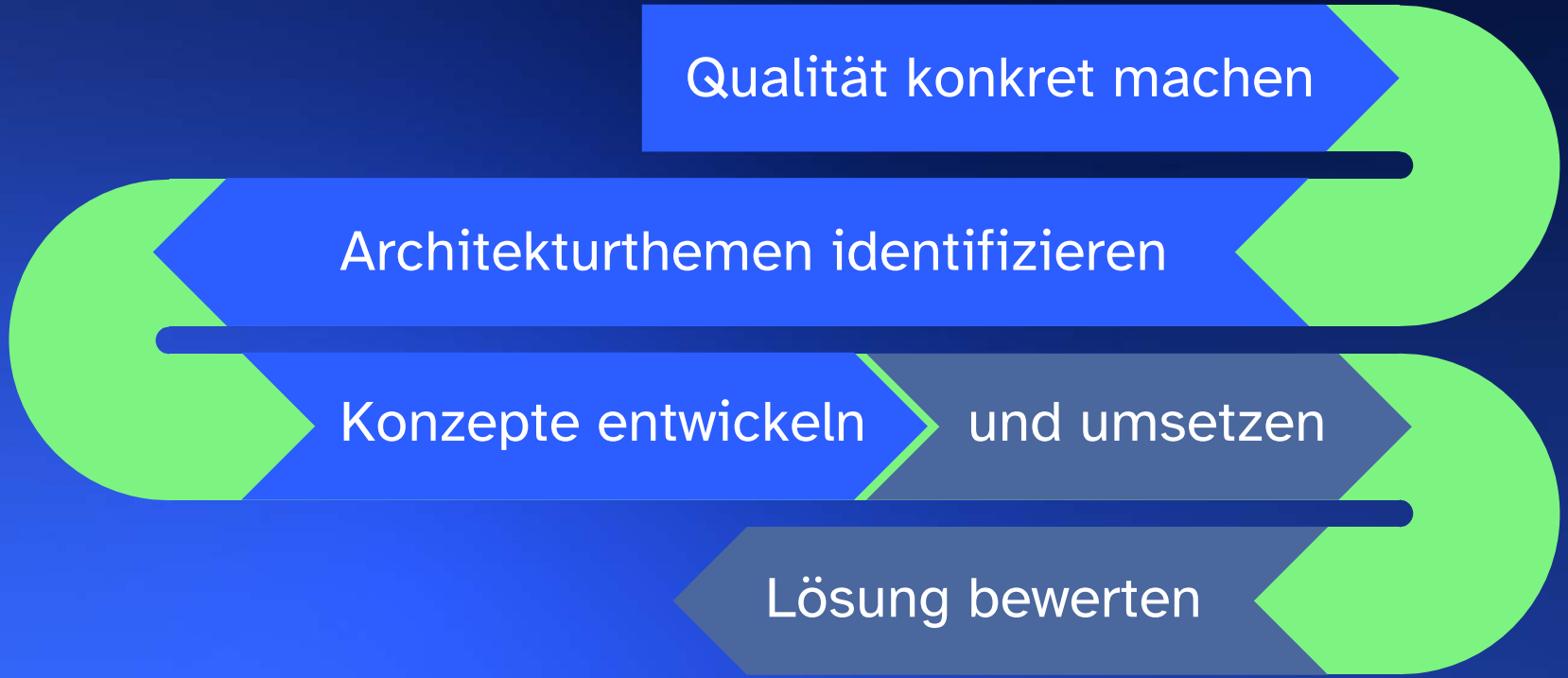
inovex

# Qualität methodisch erreichen



inovex

# Qualität methodisch erreichen



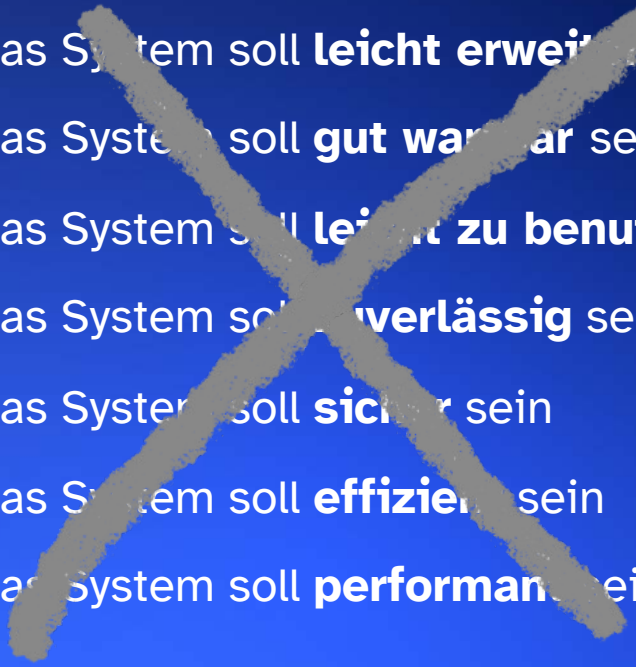
inovex

# Qualität konkret machen

Was bedeutet „Qualität“ für ein bestimmtes Produkt?

inovex

# Qualität konkret machen



- Das System soll **leicht erweiterbar** sein
- Das System soll **gut wartbar** sein
- Das System soll **leicht zu benutzen** sein
- Das System soll **zuverlässig** sein
- Das System soll **sicher** sein
- Das System soll **effizient** sein
- Das System soll **performant** sein

inovex

# Das Problem mit solchen „Qualitätsanforderungen“

- Das System soll **leicht erweiterbar** sein
- Das System soll **gut wartbar** sein
- Das System soll **leicht zu benutzen** sein
- Das System soll **zuverlässig** sein
- Das System soll **sicher** sein
- Das System soll **effizient** sein
- Das System soll **performant** sein

💩 **Unspezifisch**

💩 **Nicht testbar**

💩 **Keine belastbare Grundlage**

**Wie geht es besser?**

inovex

# Bessere Qualitätsanforderungen

1. **Qualitätsmodelle** schaffen ein klares **Vokabular** für Qualitätsmerkmale

2. **Qualitätsszenarien** sind ein bewährtes Mittel, um **spezifische**, **messbare Qualitätsanforderungen** zu schreiben

inovex

# ISO 25010:2023
# **Product Quality** Model

**Compatibility**
- Co-existence
- Interoperability

**Performance efficiency**
- Time behavior
- Resource utilization
- Capacity

**Reliability**
- Faultlessness
- Availability
- Fault tolerance
- Recoverability

**Flexibility**
- Adaptability
- Scalability
- Installability
- Replaceability

**Functional suitability**
- Functional completeness
- Functional correctness
- Functional appropriateness

**Interaction capability**
- Appropriateness recognizability
- Learnability
- Operability
- User-error protection
- User engagement
- Inclusivity
- User assistance
- Self-descriptiveness

**Safety**
- Operational constraint
- Risk Identification
- Fail safe
- Hazard warning
- Safe integration

**Maintainability**
- Modularity
- Reusability
- Analyzability
- Modifiability
- Testability

**Security**
- Confidentiality
- Integrity
- Non-repudiation
- Accountability
- Authenticity
- Resistance

inovex

# Qualitätsszenarien

Grundprinzip: **Stimulus - Response**

Es passiert etwas in der Umgebung des Systems
→ **Stimulus**

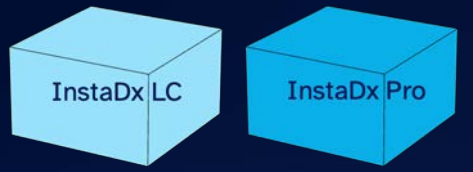
Das System reagiert darauf
→ **Response**

Die Antwort lässt sich vermessen
→ **Response Measure**

inovex

# Qualitätsszenarien – allgemeines Schema

inovex




# Beispiel: Performance

Eine Pflegekraft ruft die Ergebnis-Historie auf. Die Liste der ersten 50 Ergebnisse wird innerhalb von 100 ms angezeigt.

inovex

InstaDx Pro

# Beispiel: Reliability

Die Installation eines Software-Updates schlägt fehl. Das System startet mit dem vorherigen Softwarestand, alle Daten sind unverändert vorhanden.

inovex

InstaDx Pro

# Beispiel: Reliability

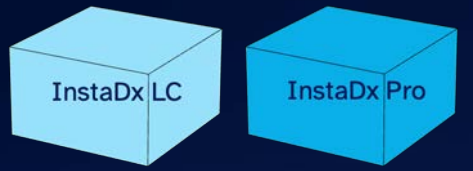
In der Device-Analytics-Funktion tritt während der Durchführung eines Tests ein kritischer Fehler auf. Das System protokolliert den Fehler und startet die Funktion neu; der Test wird erfolgreich beendet.

inovex


# Beispiel: Maintainability

Product Management will einen weiteren medizinischen Test auf den Markt bringen. Die Änderung wird als Erweiterung umgesetzt; bestehender Quellcode muss nicht geändert werden und bereits vorhandene medizinische Tests müssen nicht revalidiert werden.

inovex




# Beispiel: Portability

Der verwendete Microcontroller wird durch ein anderes Modell ersetzt. Die Software wird portiert, ohne dass Änderungen an der Applikations-Software erforderlich sind.


*Mögliche Gründe: abgekündigt, günstigeres Modell, Supply-Chain Probleme, Second Source, ...*

inovex

# Architekturthemen identifizieren

inovex


## Mission Statement

InstaDx LC is an affordable device for **fast**, **precise**, and **cost efficient** medical point of care tests.

**Architektur ist vor allem durch knappe Ressourcen getrieben**

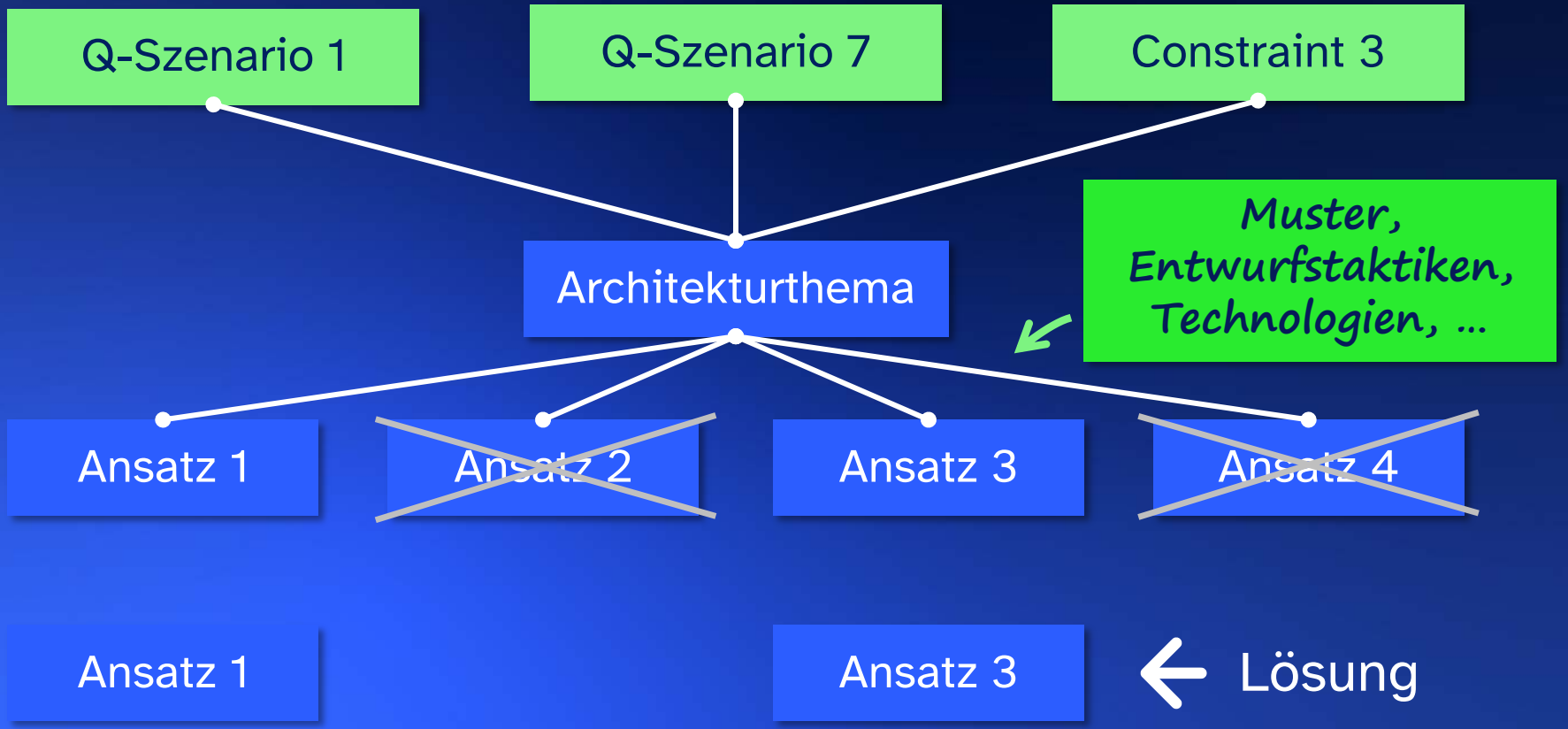

## Mission Statement

InstaDx Pro is a flexible **connected platform** for medical point of care tests that are **fast**, **precise**, and **cost efficient**.

**Architektur ist vor allem durch Flexibilität getrieben**

inovex

# Problemraum → Lösungsraum



inovex


# Fehlertoleranz

Beispiel Qualitätsszenario:

**Source**

Device-Analytics Funktion

**Stimulus**

Kritischer Fehler

**Artifact**

Gesamtsystem

**Environment**

Normaler Betrieb, Testdurchführung

**Response**

Fehler wird protokolliert, Device-Analytics Funktion neu gestartet

**Response Measure**

Test wird erfolgreich beendet

inovex


# Fehlertoleranz

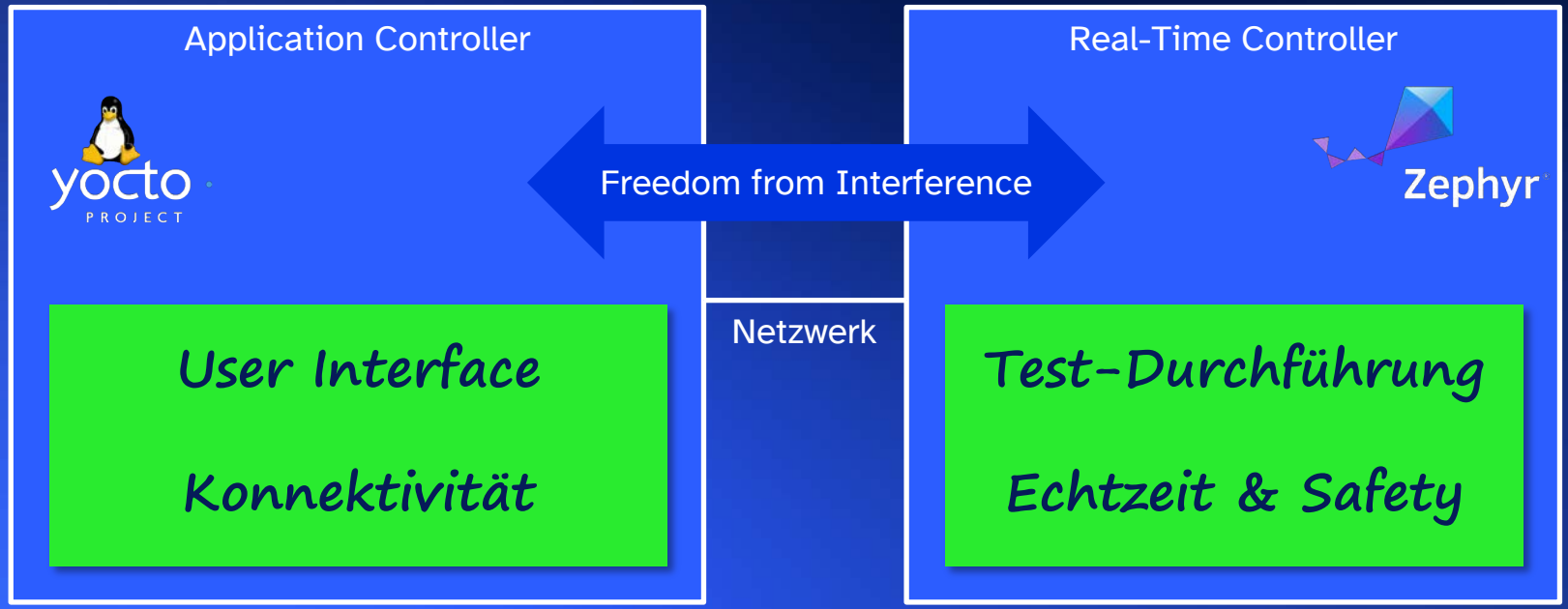
Verallgemeinert:

- Tests sollen möglichst immer durchführbar sein und ein begonnener Test erfolgreich zu Ende gebracht werden.
- Insbesondere sollen Funktionen, die bei der Testdurchführung nicht involviert sind, keinen negativen Einfluss auf die Verfügbarkeit der Testdurchführung haben.

inovex

InstaDx Pro

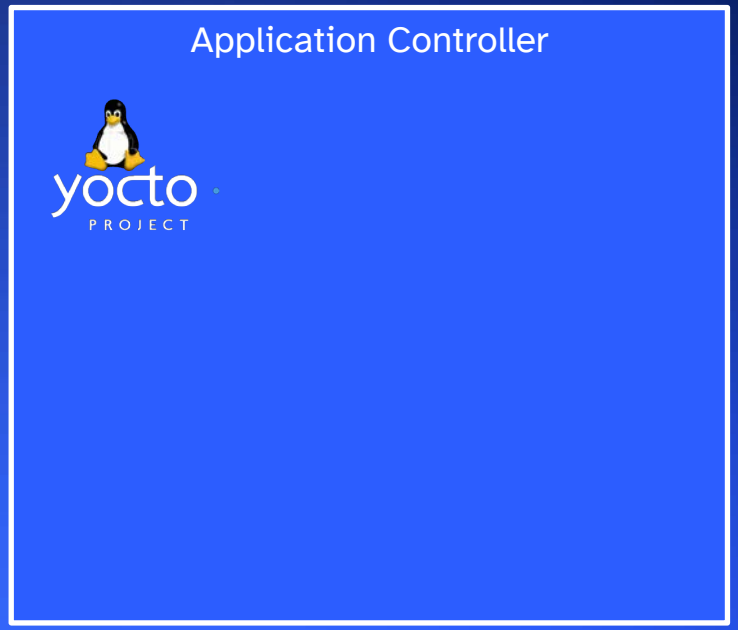
# Fehlertoleranz:
# **Segregierte System-Architektur**



inovex

InstaDx Pro

# Fehlertoleranz:
# **Multi-Prozess-Architektur**

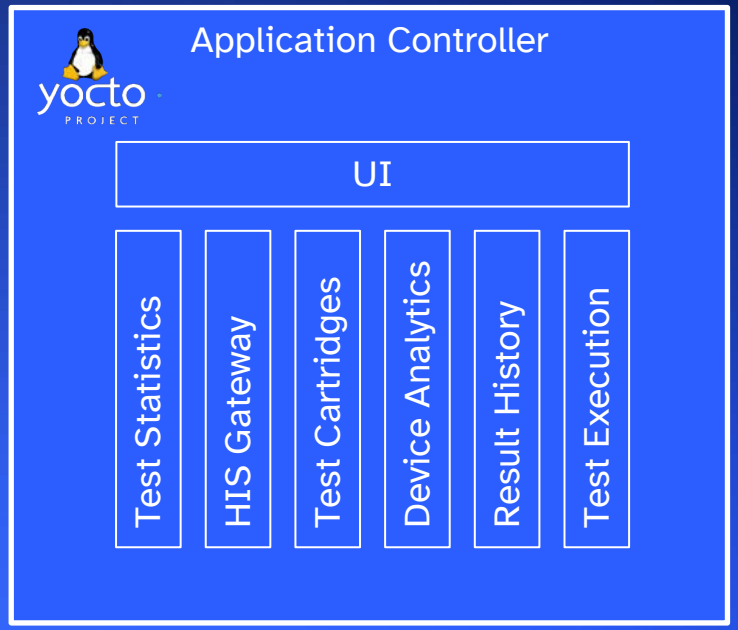



- Test Statistics
- HIS Gateway
- Test Cartridges
- Device Analytics
- Result History
- Test Execution

inovex

InstaDx Pro

# Fehlertoleranz:
# **Multi-Prozess-Architektur**



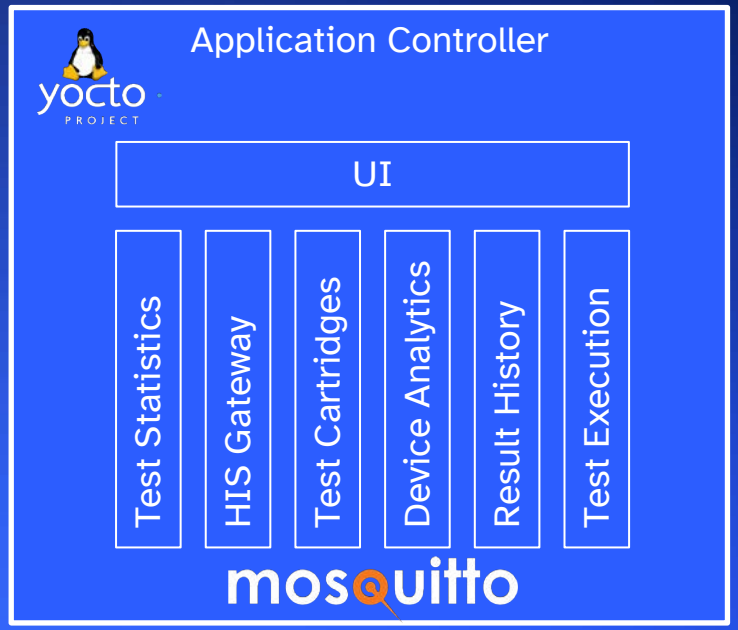

Linux Prozesse als „Units of Mitigation“ innerhalb des Application Controllers

Lifecycle-Management über systemd (Erkennung von crash failures und silent failures, Restart)

inovex

InstaDx Pro

# Fehlertoleranz:
# Lose Kopplung – asynchrones Messaging

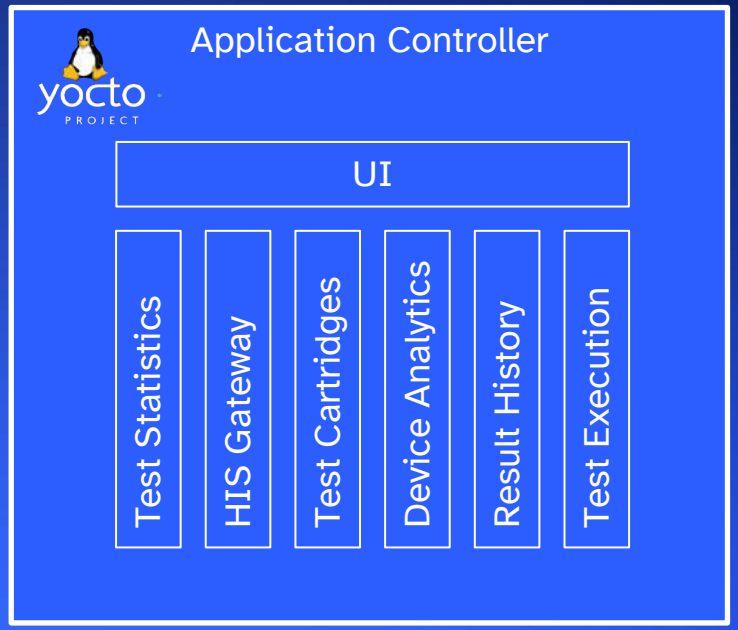



Bei der Test-Durchführung nicht involvierte Services wie Test Statistics, Device Analytics, HIS Gateway beziehen ihren Input asynchron via publish-subscribe

inovex

InstaDx Pro

# Fehlertoleranz:
# **Expect Errors – Retry**




Umsetzung von Retry-Mechanismen (mit Limits) bei der Interprozess-Kommunikation

inovex

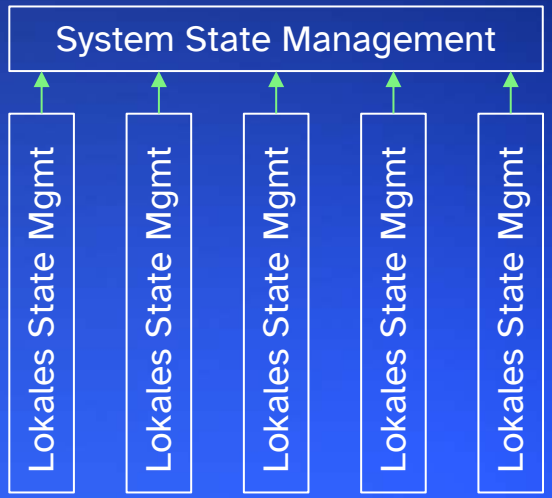
InstaDx Pro

# Fehlertoleranz:
# **Zeitliche Redundanz**

inovex

InstaDx Pro

# Fehlertoleranz:
# **Zustands-Management & Eskalation**




Verwaltung von Betriebs-Zuständen pro Service

Falls lokale Behebung erfolglos: Eskalation

inovex


# Zusammenfassung: Fehlertoleranz

Zephyr

(yocto) Linux

Segregierte Systemarchitektur

Multi-Process Architecture

Zeitliche Entkopplung

Services als „Unit of Mitgation“

Service Lifecycle Management

Asynchrones Messaging

MQTT

Operational State Management

Zeitliche Redundanz

inovex


# Thema: Fehlertoleranz

Real-Time Controller

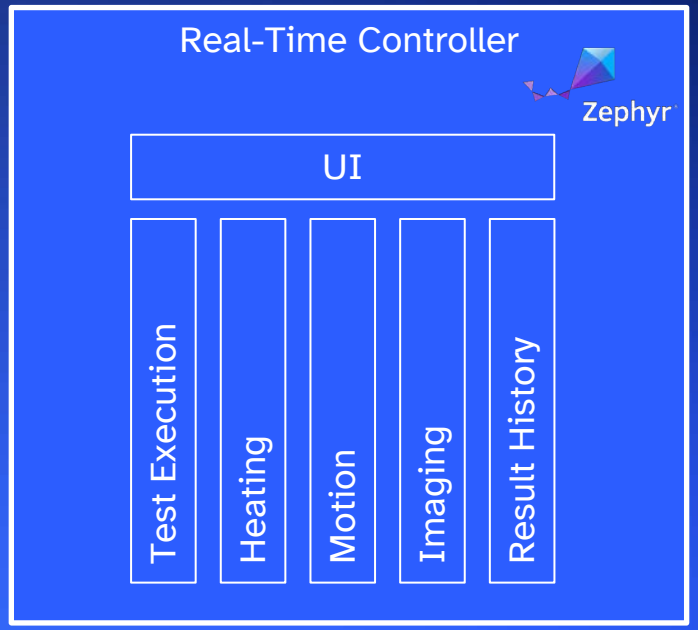
Zephyr

Keine segregierte System-Architektur

Alles (UI, Backend-Services, eigentliche Test-Ausführung) läuft auf einem Microcontroller

inovex


# Thema: Fehlertoleranz

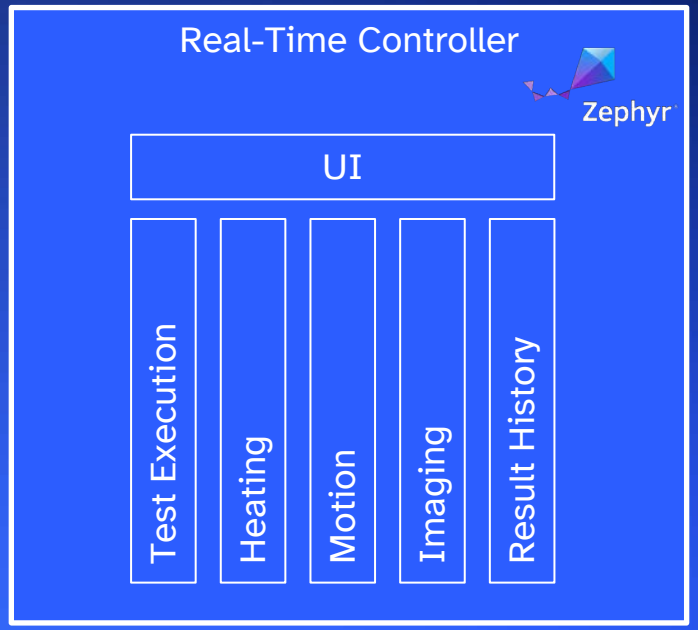



Keine segregierte System-Architektur

Alles (UI, Backend-Services, eigentliche Test-Ausführung) läuft auf einem Microcontroller

inovex


# Thema: Fehlertoleranz




Keine Prozess-Isolation

Bei kritischen Fehlern:

Das gesamte System neu starten

inovex

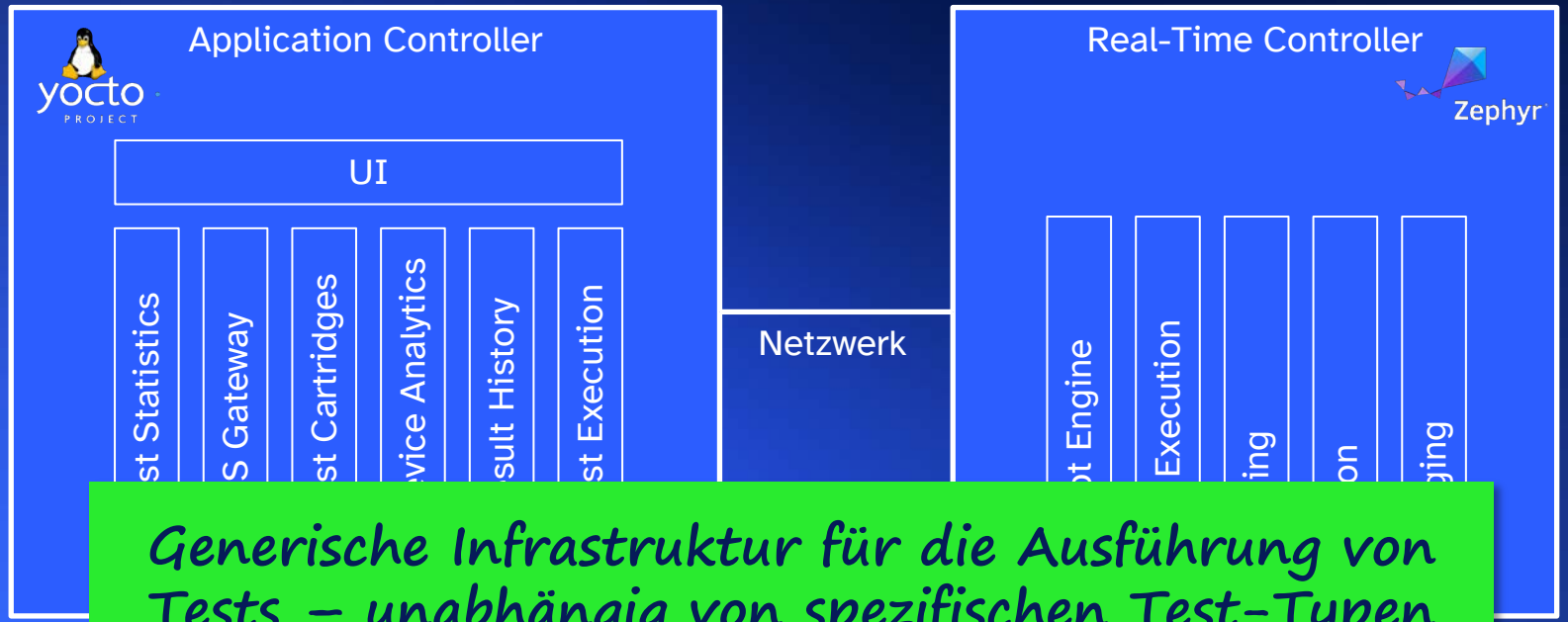

# Thema: Erweiterbarkeit

Beispiel Qualitätsszenario:

inovex

InstaDx Pro

# Erweiterbarkeit:
# **Neuer Test als Erweiterung** (Open-Closed Principle)

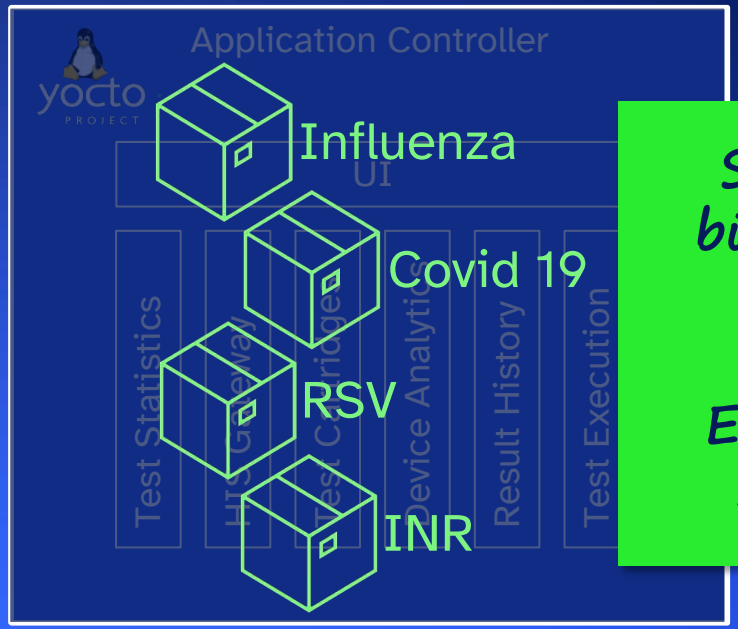


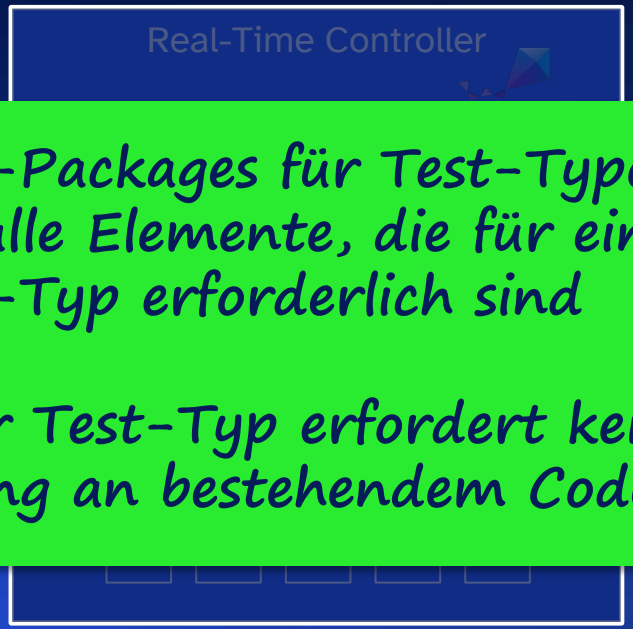

*Generische Infrastruktur für die Ausführung von Tests – unabhängig von spezifischen Test-Typen*

inovex

InstaDx Pro

# Erweiterbarkeit:
# **Neuer Test als Erweiterung** (Open-Closed Principle)



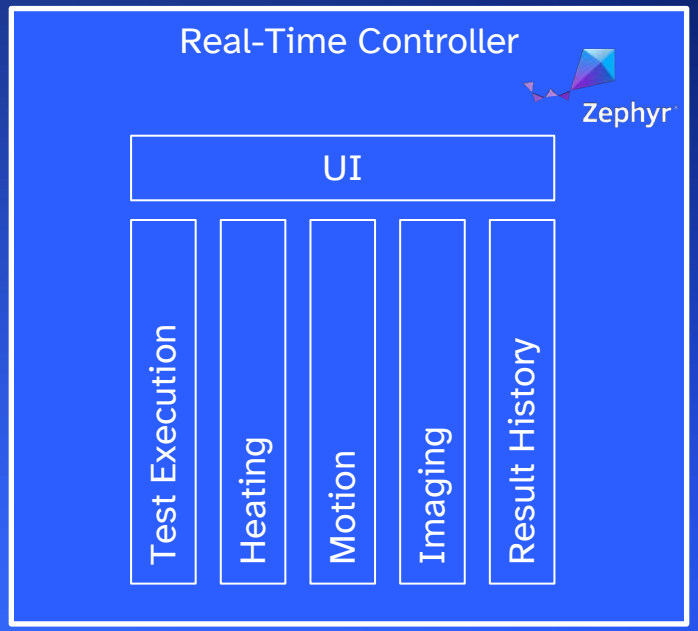




*Support-Packages für Test-Typen bündeln alle Elemente, die für einen Test-Typ erforderlich sind*

*Ein neuer Test-Typ erfordert keine Änderung an bestehendem Code.*

inovex

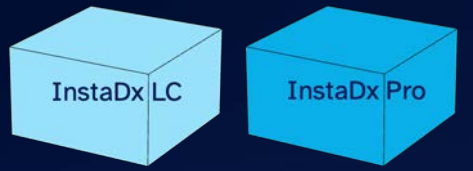

# Thema: Erweiterbarkeit




Keine Trennung zwischen test-spezifischen und generischen Anteilen.

Neuer Test-Typ erfordert Änderungen an bestehendem Code – Aufwände für Verifikation!

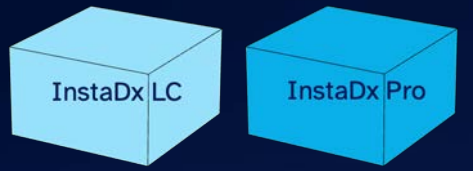
inovex




# Portierbarkeit

Beispiel Qualitätsszenario:

inovex




# Thema: Portabilität

Adressiert durch Betriebssystem-Auswahl:



- Linux und Zephyr bieten breite **Hardware-Unterstützung** und gute **Hardware-Abstraktion**
- **Devicetree** ermöglicht Entkopplung zwischen Applikations-Software und Hardware-Details
- Anwendungssoftware abstrahiert von Hardware-Spezifika

inovex

# Architektur wird durch Anforderungen getrieben

inovex

# Architektur wird durch Anforderungen getrieben

## Qualitätsanforderungen und Randbedingungen haben prägenden Einfluss

inovex

**Architektur wird durch Anforderungen getrieben**

**Qualitätsanforderungen und Randbedingungen haben prägenden Einfluss**

**Architektur systematisch ableiten**
**→ Geschäftsziele unterstützen**

# Vielen Dank!

**Ulrich Becker**
Software Architect

+49 174 / 7065171
ulrich.becker@inovex.de