Software-Architektur

Qualitätsmerkmal orientierte SW-Architektur
durch Anwendung von *Architectural Styles*

Markus Pauls
markus.pauls@brightONE.de

Inhalt

- Vorstellung
- Was ist Software-Architektur?
  - Qualitätsmerkmale
  - Szenarien
- Was sind *Architectural Styles*?
- Projektbeispiele
- Zusammenfassung

Vorstellung

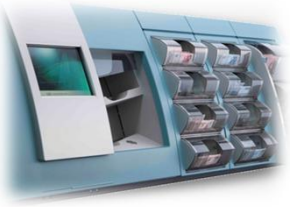
brightONE Embedded Systems

Smart Products for a Connected World

Smart Products for a Connected World

Automotive

Hi-Tech Electronics



Medical

Human-Machine-Interface Solutions

M2M- & High-Speed-Connectivity Solutions

Functional Safety Solutions

Concepts & Innovation → **Design & Development** → **Lifecycle Services**

Was ist SW-Architektur?

Was ist SW-Architektur?

- ***Software-Architektur*** bedeutet das Zerlegen eines Gesamtsystems in ***Komponenten*** und eine Beschreibung der Komponenten, sowie der ***Schnittstellen*** und der ***Beziehungen*** dieser Komponenten untereinander.

  → „Teile und herrsche“ Strategie

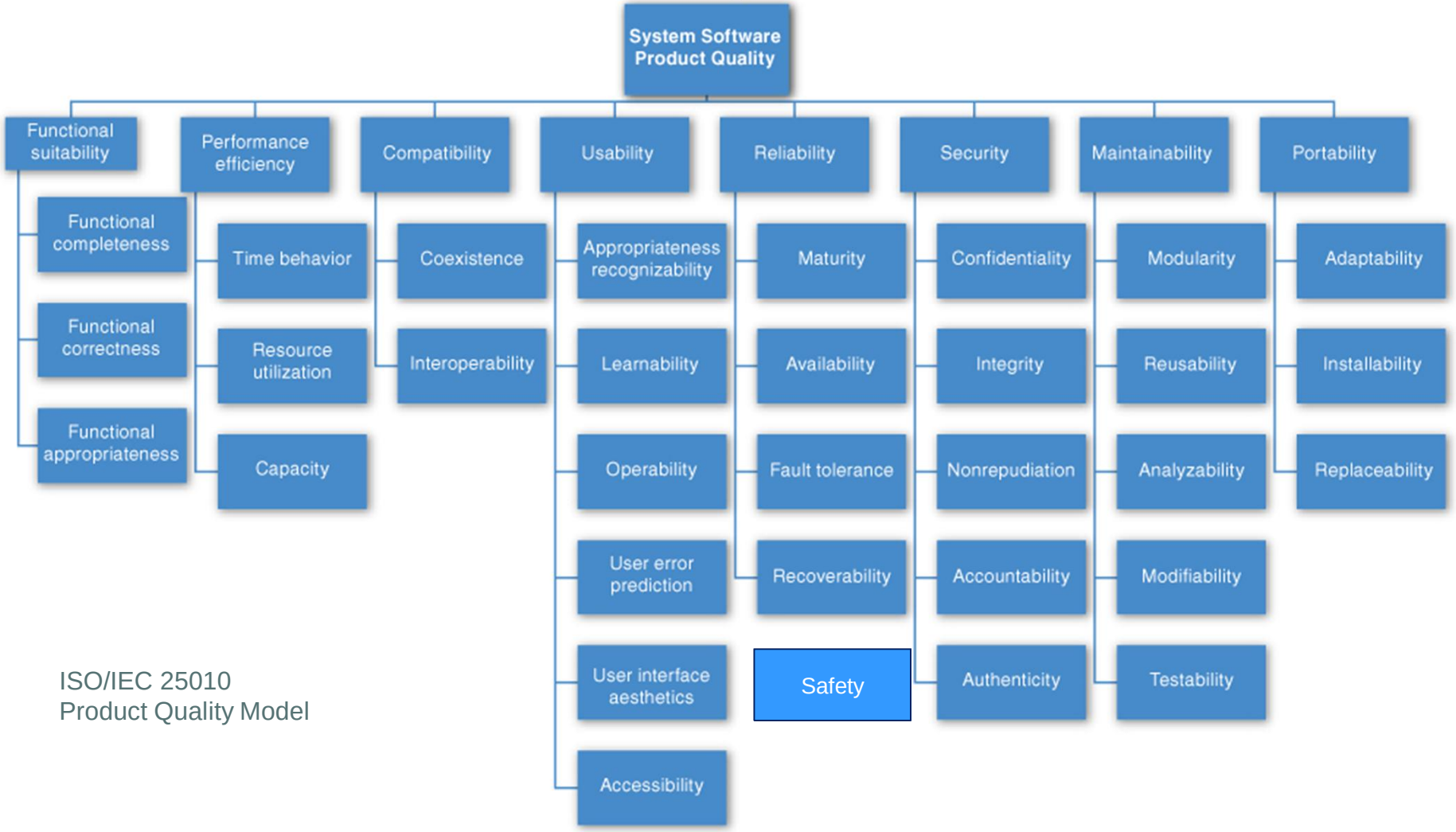
- Software-Architektur ist das Ergebnis von weitreichenden ***Design-Entscheidungen***, die notwendig sind bevor eine Gruppe von Personen gemeinsam ein Software-System erstellen können.

- Die **Design-Entscheidungen** werden im wesentlichen durch ***Qualitätsmerkmale*** bestimmt.

Was ist SW-Architektur?

- Kategorien von Design-Entscheidungen:
  1. Zuweisung von Verantwortlichkeiten
  2. Koordinations-Modell
  3. Daten-Modell
  4. Ressourcen-Management
  5. Abbildung zwischen Architektur-Elementen
  6. Binde-Zeit-Punkt
  7. Wahl der Technologie

Qualitätsmerkmale
== „non functional requirements“



ISO/IEC 25010
Product Quality Model

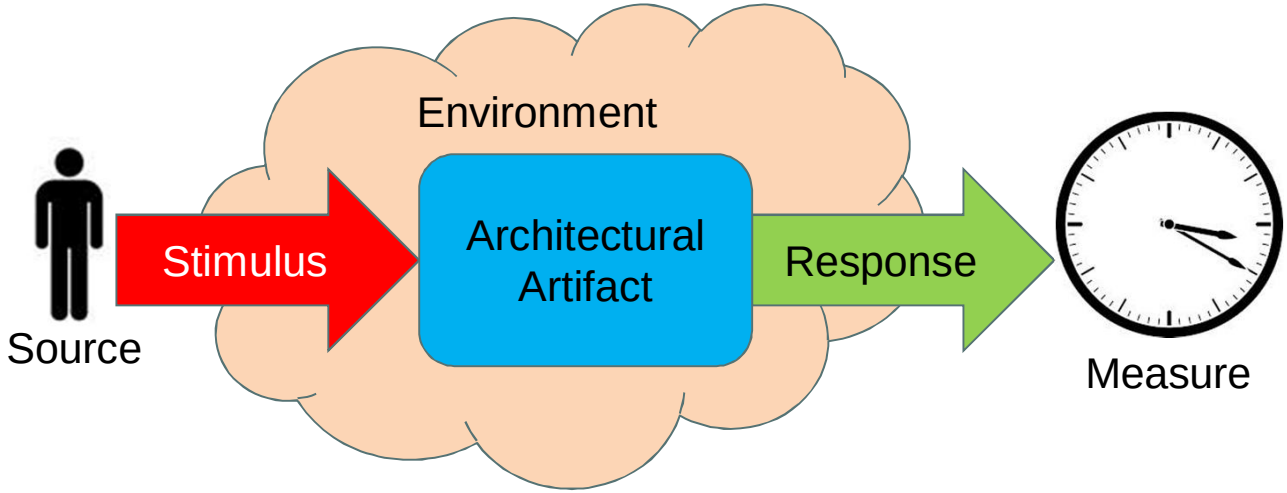
Bewertung von Architekturen → Szenarien

- *Non-functional requirements* häufig schlecht quantifizierbar:
  - "*The system shall be robust.*"
  - "*The system shall be highly modifiable.*"
  - "*The system shall be secure from unauthorized break-in.*"
  - *"The system shall exhibit acceptable performance."*

- Bewertung mit Hilfe von *Szenarien*:
  - Uses Cases → typische Anwendungen
  - Growth scenarious → typische zukünftige Änderungen
  - Exploratory scenarious → extreme Änderungen, „Stresstest"

- Szenarien helfen Fragen an die Architektur des System zu stellen.

Bewertung von Architekturen
→ Formalisierung von Szenarien



- ***Stimulus***: Das Ereignis auf das die Architektur reagieren muss.
- ***Source***: Quelle des Stimulus
- ***Environment***: Satz von Randbedingungen unter dem das Szenarium stattfindet.
- ***Architectural Artifact***: Der Teil der Architektur der auf das Ereignis reagiert.
- ***Response***: Ein messbare (oder zumindest beobachtbare) Reaktion auf den Stimulus.
- ***Measure***: Akzeptanz-Kriterium für die Reaktion

Taxonomie
Performance Szenarien

Source	Stimulus	Environment	Architectural Artifact	Response	Measure
Internal …	Periodic …	Normal …	System	Process events	Latency
External …	Sporadic …	Overload …	Component	Change level of service	Deadline
	Bursty …	Reduced capacity …			Throughput
	Stochastic …	Emergency …			Jitter
		Peak …			Miss rate
					Data loss
… to the system	… events	… mode			

Taxonomie Modifizierbarkeit‘s Szenarien

Source	Stimulus	Environment	Architectural Artifact	Response	Measure
End-user	Add …	At Runtime	Code	Make …	Cost in effort
Developer	Delete …	At Compile Time	Data	Test …	Cost in money
System Administrator	Modify …	At Build Time	Interfaces	Deploy …	Cost in time
		Design Time	Components		Cost in number, size, complexity of affected artifacts
		Initiation Time	Resources		Extent affects other system functions or qualities
			Configurations		New defects introduced
	… functionality, quality attribute, capacity or technology			… modification	

Sensitivity and tradeoff points

Diese Begriffe charakterisieren Schlüssel-Entscheidungen:

- A ***sensitivity point*** is a property of one or more components (and/or component relationships) that is critical for achieving a particular quality attribute.
- Sensitivity points tell you where to focus your attention when trying to achieve a quality goal. They serve as yellow flags: “Use caution when changing this property of the architecture”.

- A **trade-off** is a situation that involves losing one quality of something in return for gaining another quality.
- A ***trade-off point*** is a property that affects more than one quality attribute and is a sensitivity point for at least one attribute.

Was sind *Architectural Styles*?

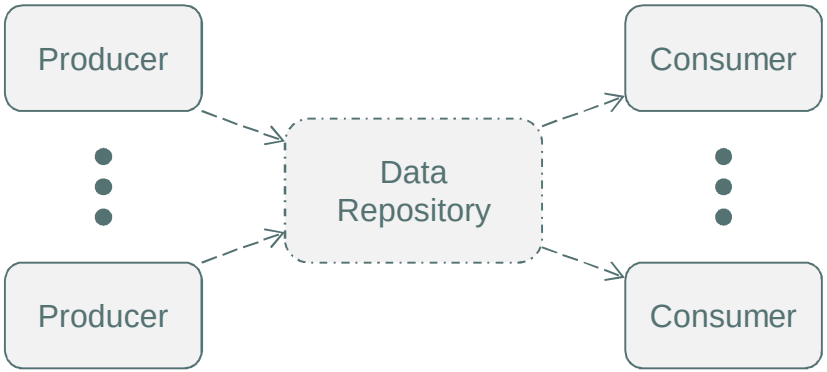
Architectural Styles

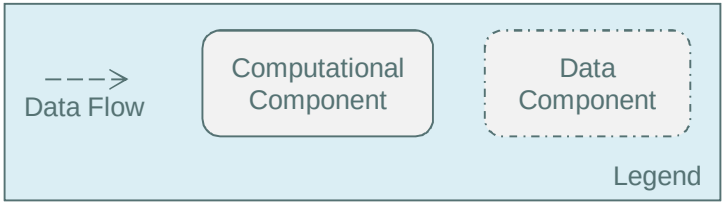
- Ziel: Das Rad nicht jedes mal neu erfinden.

  → Verwendung von **Lösungs-Mustern** mit bekannten Vor- und Nachteilen.

- „An ***architectural style*** is a description of ***component types*** and their ***topology***, which includes a description of the pattern of data and control interaction among the components. Architectural styles also provide an informal description of the benefits and drawbacks of using that style.”

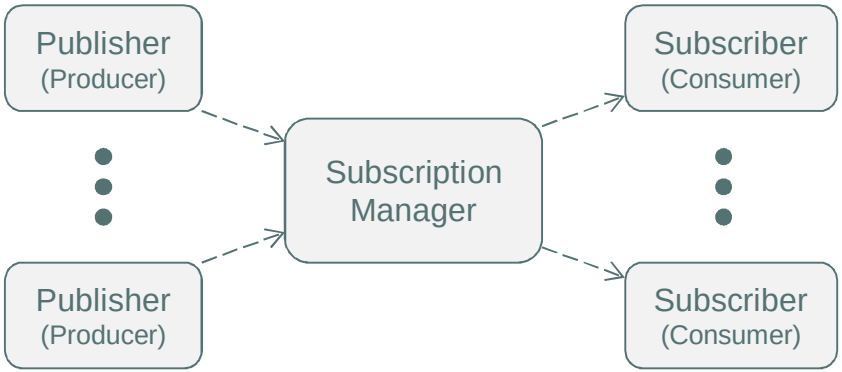
Beispiel:
Data Indirection – Abstract Data Repository

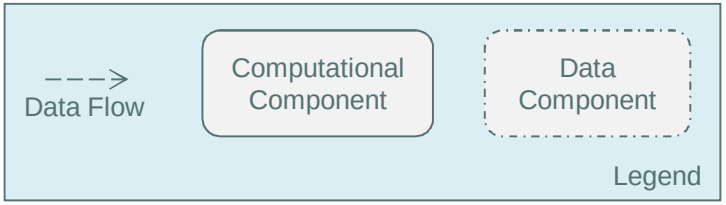




- Ziel:
  Verbesserung der Modifizierbarkeit durch Verringerung der Datenfluss- und Kontroll-Beziehungen zwischen den Komponenten.
- Charakteristika:
  - Topologie: Stern
  - Daten-Persistenz: persistent
  - Kenntnis der Datenstruktur im Client: vollständige Kenntnis
  - Aktivität: passiv
- Producer und Consumer können leicht ausgetauscht werden.
- Änderung der Daten-Strukturen problematisch.
- Synchronisation der Schreib- und Leseraten evtl. erforderlich.

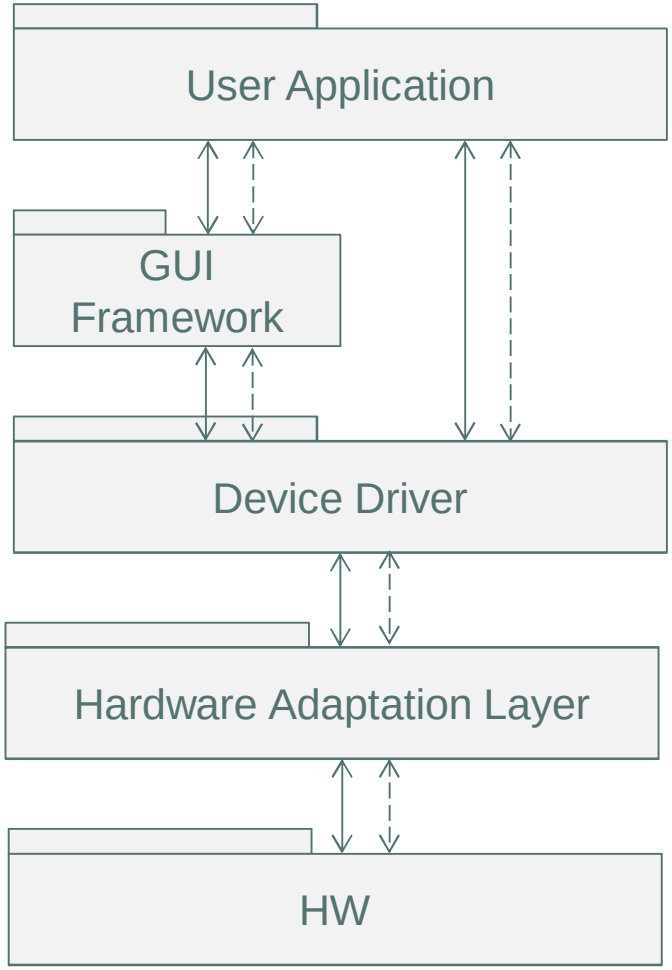
Beispiel:
Data Indirection – Publish/Subscribe





- Ziel:
  Verbesserung der Modifizierbarkeit durch Verringerung der Datenfluss- und Kontroll-Beziehungen zwischen den Komponenten.
- Charakteristika:
  - Topologie: Stern
  - Daten-Persistenz: transient
  - Kenntnis der Datenstruktur im Client: keine Kenntnis
  - Aktivität: Aktiv
- Publisher und Subscriber können leicht ausgetauscht werden.
- Schreib- und Leseraten automatisch synchronisiert.
- Keine Aussage darüber wann der Subscriber informiert wird.

Beispiel: Layering



- Ziel:
  Verbesserung der Wartbarkeit und Portierbarkeit.

- Charakteristika:
  - Topologie: Schichten
  - Abhängigkeiten:
    - strikt ⇔ locker
    - Downwards only
      ⇔ Downwards and Upwards
  - Kenntnis der Datenstrukturen: keine Kenntnis

- Schichten können leicht ausgetauscht werden.
- Negativer Einfluss auf Performance.
- Lockere Schichtung oder Abhängigkeiten in beiden Richtungen kann dem Ziel entgegen wirken.

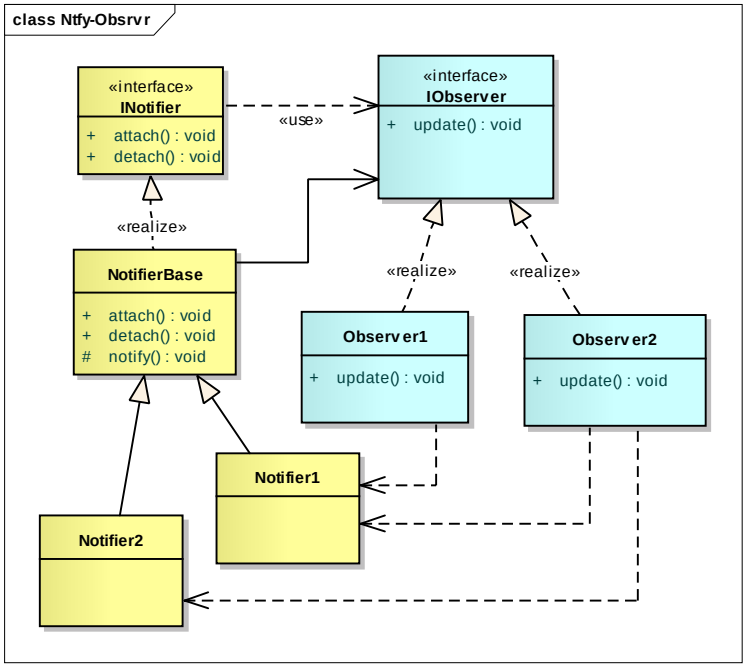
Projektbeispiele

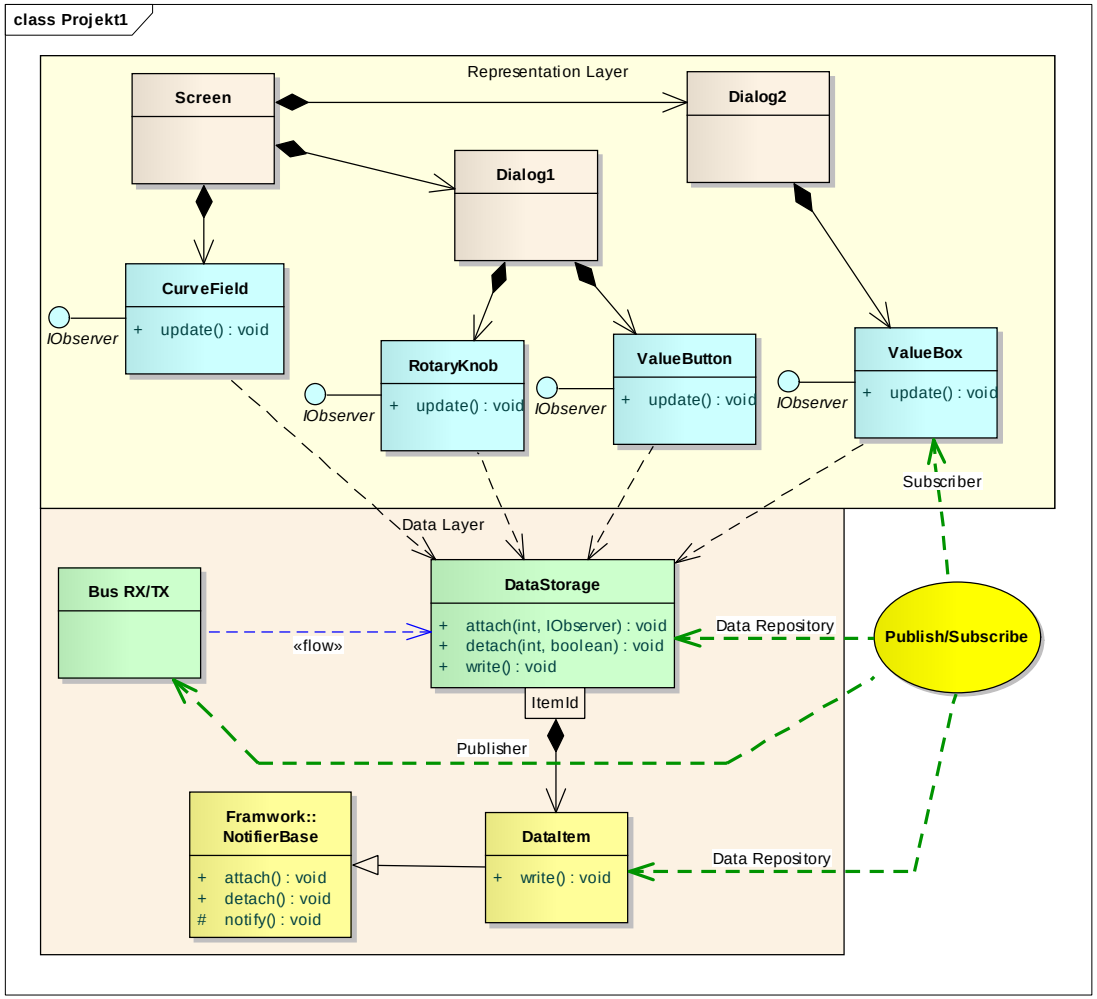
Projektbeispiele

1. Display- und Bedien-Einheit eines Intensiv-Medizinischen Gerätes
   - Neuentwicklung der Display- und Bedien-Einheit.
   - Wiederverwendung der bestehenden Gerätesoftware mit nur geringfügigen Änderungen.
   - Kopplung Backend ⇔ Display Einheit über einen Feldbus.
   - RTOS mit monolithischem Speichermodell

2. Notfall-Medizinisches Gerät mit Reanimations- und Überwachungsfunktion
   - Neuentwicklung von Elektronik und **SW** der ‚Steuerplattform'.
     - Hardware nahe SW Anteile (BSP, Treiber)
     - Objektorientierte Applikationssoftware
       - ↳ Steuerung verschiedener zugelieferter Sensor- und Aktuator-Module
     - Graphische Benutzeroberfläche
     - POSIX-RTOS ⇨ Multi-Prozess-Environment

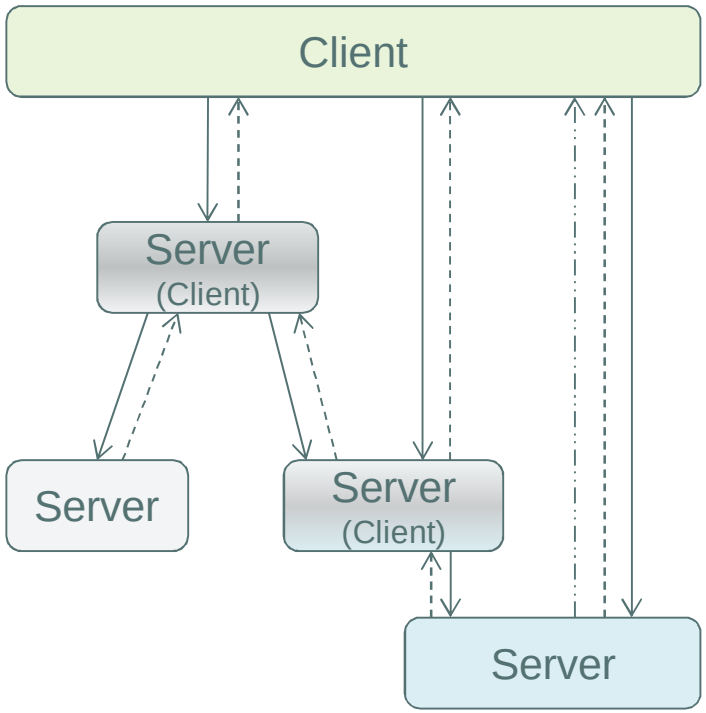
Data Indirection:
Notify/Oberver Pattern (Projekt 1)

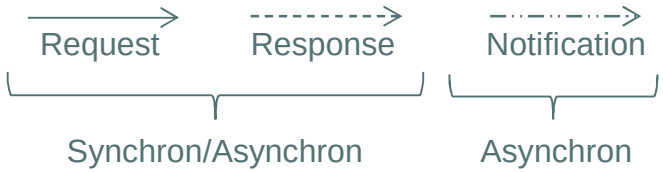




- Objektorientierte Umsetzung von Publish/Subscribe

Distributed Systems:
Client/Server

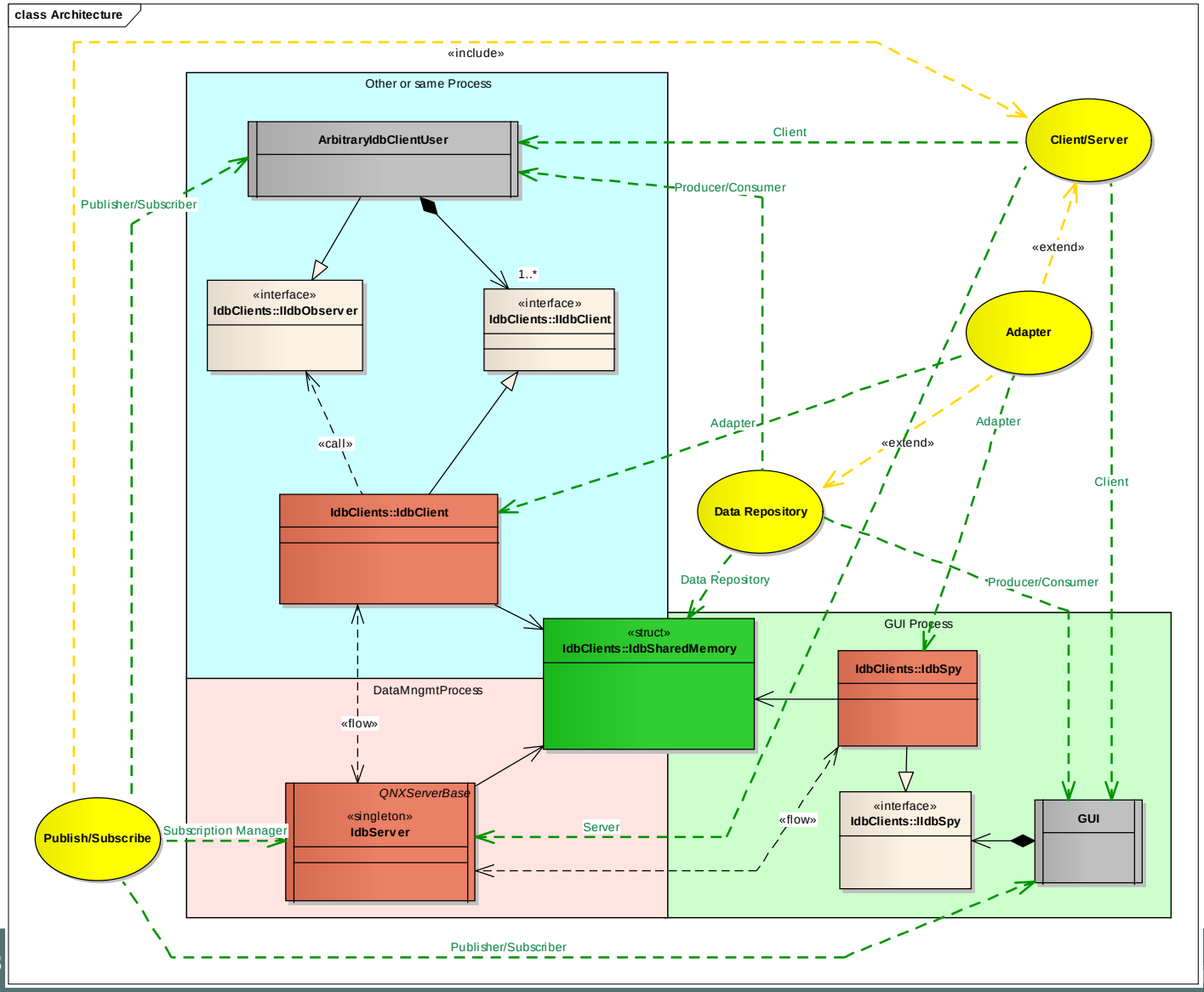




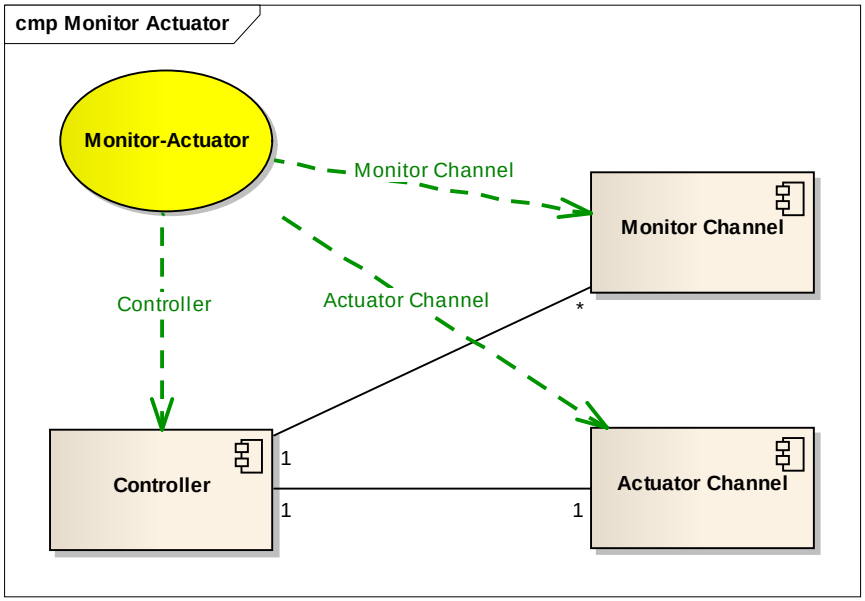
- Partitions tasks or workloads between the providers of a resource or service, called ***servers***, and service requesters, called ***clients***.
  - Often clients and servers communicate over a computer network on separate hardware, but both client and server may reside in the same system.
- Clients and servers exchange messages in a ***request-response*** **messaging pattern**:
  - The client sends a request, and the server returns a response.
- Directed communication
- Notification requires a ***stateful*** **server**

Data Indirection

Publish/Subscribe (Projekt 2)



Safety and Reliability:
Monitor-Actuator



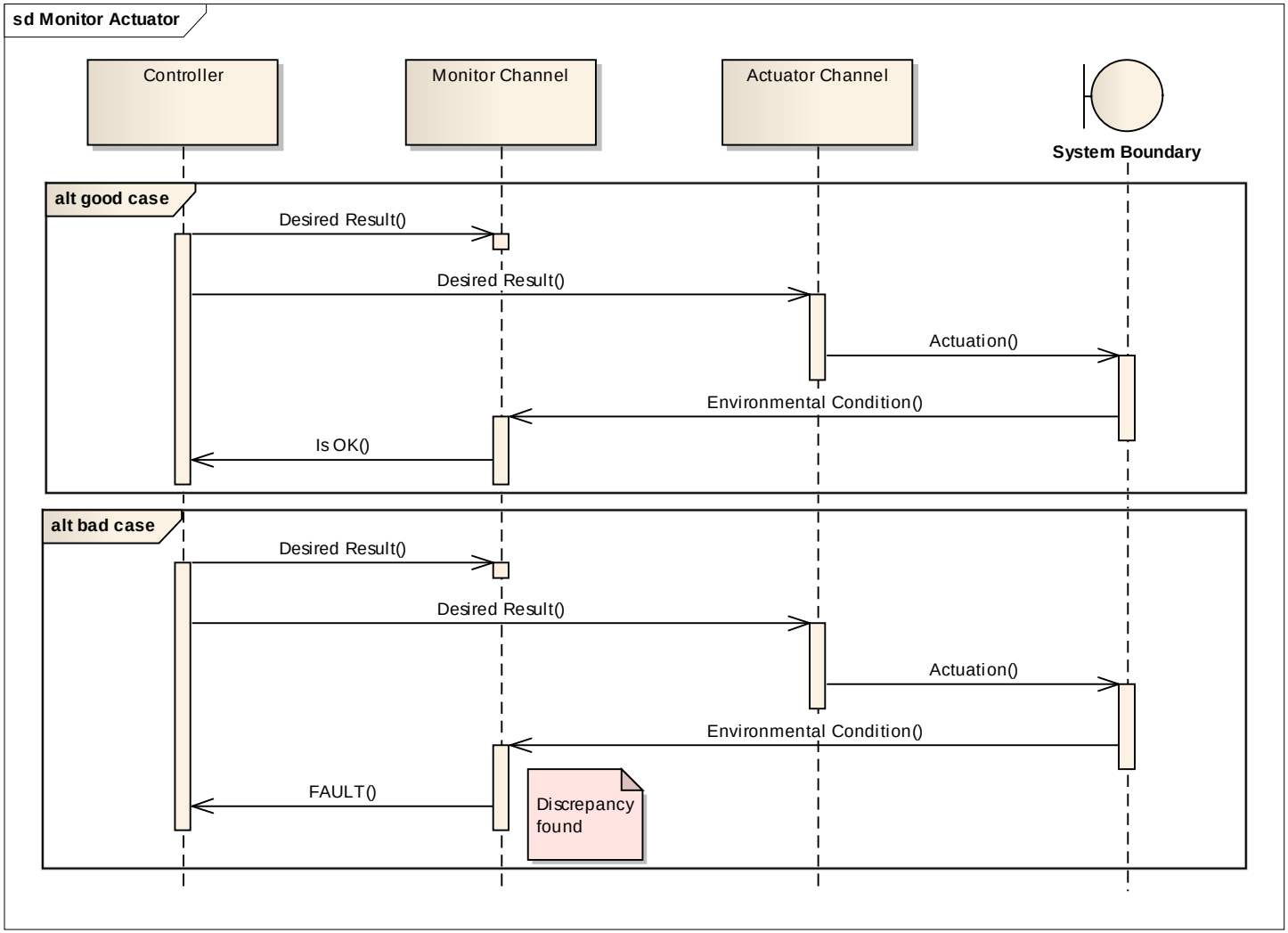
- Problem:
  Detect faults and provide corrective actions in the presence of faults.

- Solution:
  - Use a primary heavyweight channel providing the full functionality.
  - Use a second lightweight channel that detect faults within the primary by checking the results of the actuation.

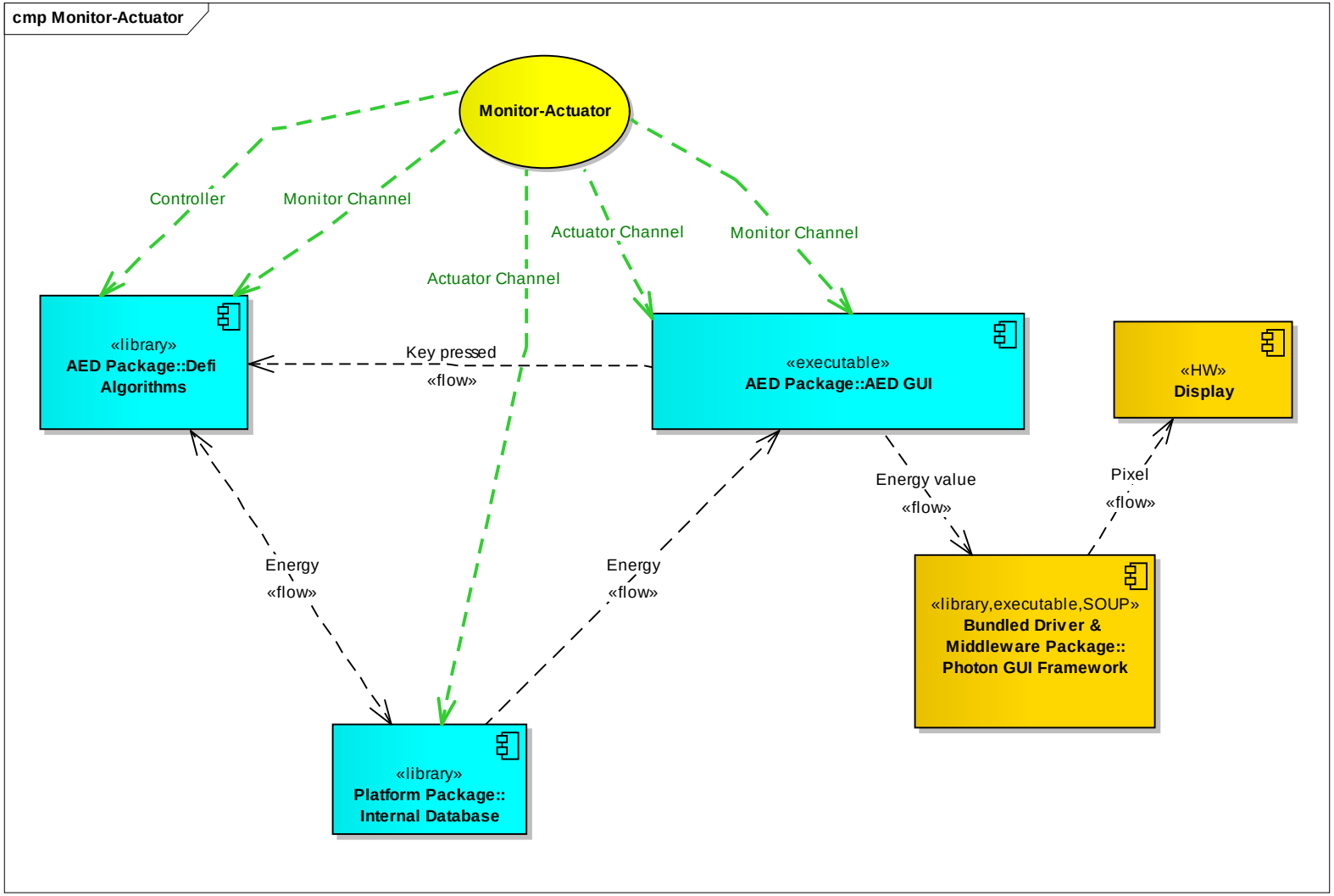
- Definition:
  A ***channel*** is a collaboration of objects which performs a series of safety-relevant computations or actuations.

Safety and Reliability:

Monitor-Actuator



Safety and Reliability:
Monitor-Actuator (Projekt 2)



Zusammenfassung

1. Software-Architektur ist das Ergebnis von ***Design-Entscheidungen***.
2. Software-Architektur bestimmt in erheblichem Umfang die erreichbaren ***Qualitätsmerkmale*** eines Systems.
3. Szenarien präzisieren Qualitätsmerkmale.
4. Szenarien helfen Software-Architekturen zu bewerten.
5. ***Architectural-Styles*** sind Lösungsmuster mit bekannten Vor- und Nachteilen.

Markus Pauls
markus.pauls@brightONE.de

Smart Products for a Connected World

Partner for
Innovation & Development

www.brightONE.de/SmartProducts

blog.brightONE.de/SmartProducts