

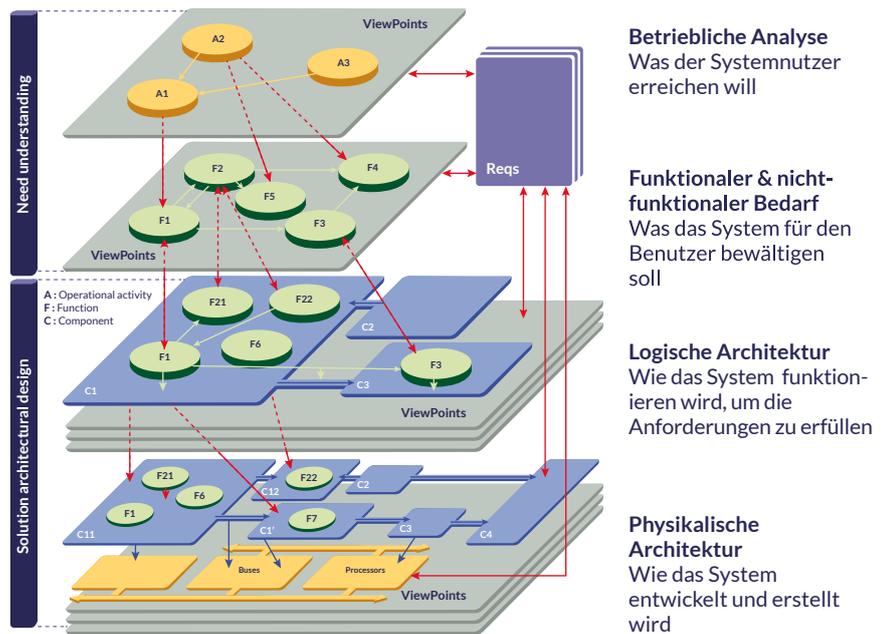
EINE SOFTWAREUNTERSTÜTZTE METHODE ZUR

# Definition, Analyse, Entwicklung & Validierung von System-, Software- und Hardware-Architektur

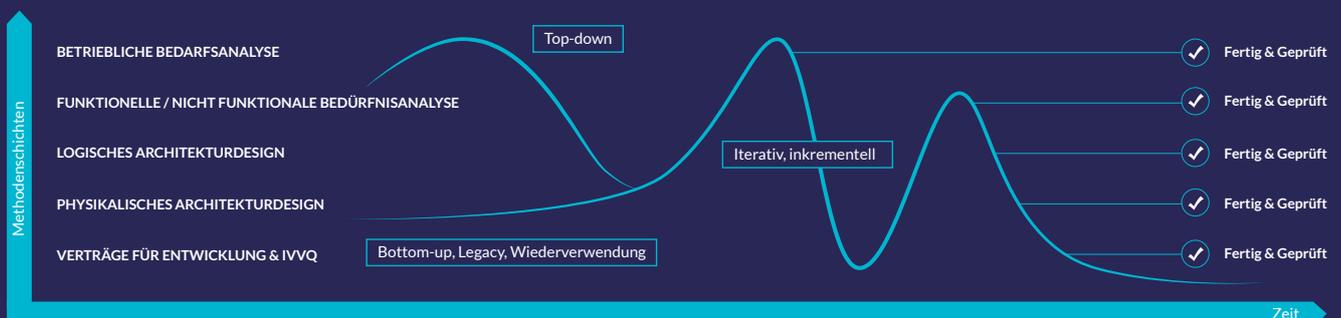
Unterstützt  
effektive Zusammenarbeit  
im Engineering



Erlaubt die Validierung einer Lösung  
gegenüber den betrieblichen Anforderungen  
Vereinfacht Auswirkungenanalysen



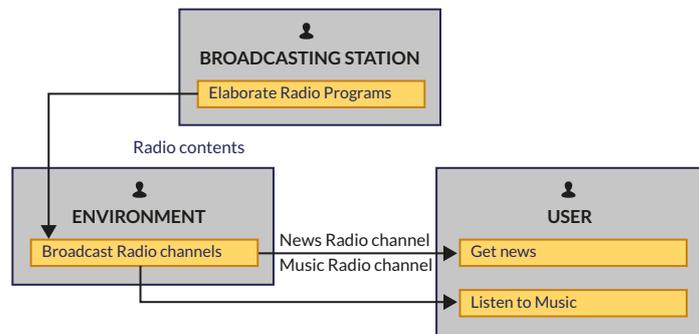
## Kompatibel mit den meisten Prozessen top-down bottom-up, Iterativ, Legacy-basiert, gemischt...



## Analyse der betrieblichen Anforderungen des Kunden

Was die Systemnutzer erreichen wollen

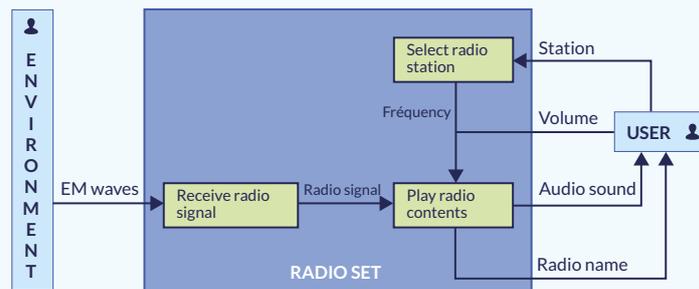
- ✓ Operational capabilities definieren
- ✓ Analyse der betrieblichen Anforderungen durchführen



## Analyse der System SW- und HW-Anforderungen

Was das System für die Nutzer leisten soll

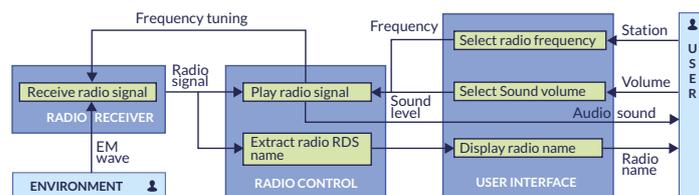
- ✓ Capability trade-off analysieren
- ✓ Funktionale und nichtfunktionale Analyse durchführen
- ✓ Anforderungen formalisieren und konsolidieren



## Design der logischen Architek

Wie das System funktionieren wird, um Anforderungen zu erfüllen

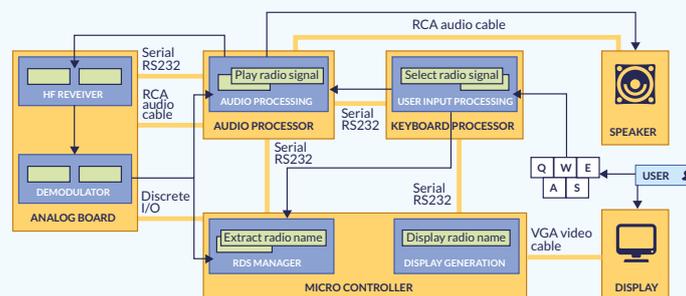
- ✓ Architekturrichtlinien und Sichten auf die Architektur definieren
- ✓ Mögliche Aufteilungen der Komponenten erstellen
- ✓ Den besten Kompromiss für die Architektur auswählen



## Design der physikalischen Architektur

Wie das System entwickelt und erstellt wird

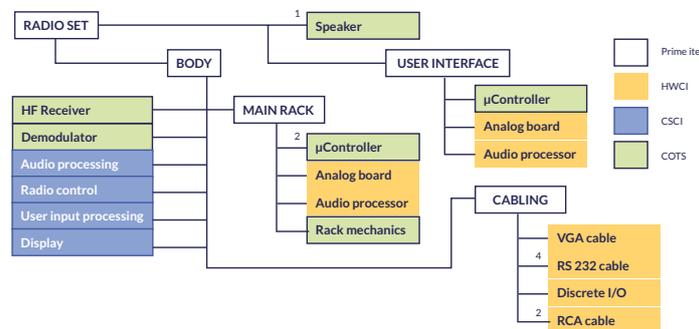
- ✓ Architekturmuster definieren
- ✓ Wiederverwendung bestehender Assets berücksichtigen
- ✓ Eine physikalische Referenzarchitektur entwickeln
- ✓ Validieren und prüfen



## Entwicklungsvereinbarungen

Was von jedem Entwickler/Zulieferer erwartet wird

- ✓ Eine Komponenten-IVVQ Strategie definieren
- ✓ Eine Vereinbarung zur Produktstruktur und Komponentenintegration definieren und umsetzen



- Operational Capabilities
- Aktoren, operational entities
- Activities
- Interaktionen zwischen Activities & Aktoren
- In Activities und Interaktionen verwendete Informationen
- Betriebliche Prozesse als Abfolge von activities
- Szenarien für dynamisches Verhalten

- Aktoren und System, Capabilities
- Funktionen des Systems & der Aktoren
- Datenaustausch zwischen Funktionen
- Datenflüsse durch Functional Chains
- In Funktionen & im Datenaustausch verwendete Informationen, Datenmodell
- Szenarien für dynamisches Verhalten
- Modes & States (Zustandsmaschinen)

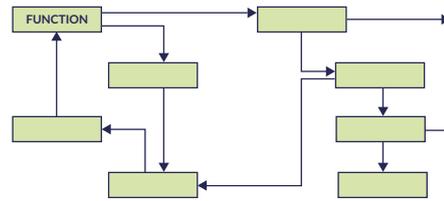
**GLEICHE KONZEPTE, ZUZÜGLICH:**

- Komponenten
- Komponentenaports und Schnittstellen
- Zuweisung der Funktionen zu den Komponenten
- Definition der Komponentenschnittstellen durch Zuweisung von Functional Exchanges

**GLEICHE KONZEPTE, ZUZÜGLICH:**

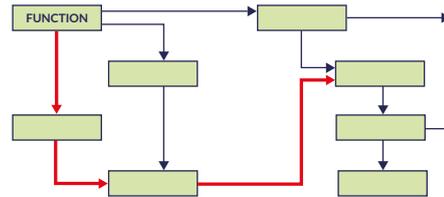
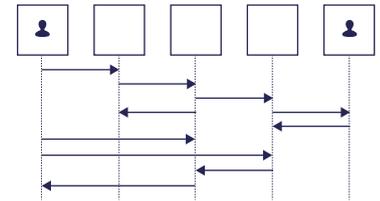
- Verhaltenskomponenten verfeinern logische Komponenten und implementieren das Funktionsverhalten
- Implementierungskomponenten stellen die Ressourcen für die Verhaltenskomponenten zur Verfügung
- Physikalische Links zwischen Implementierungskomponenten

- Baum aller Konfigurationsobjekte
- Teilenummern, Anzahl
- Entwicklungsvereinbarungen (erwartetes Verhalten, Schnittstellen, Szenarien, Ressourcenverbrauch, nichtfunktionale Eigenschaften...)

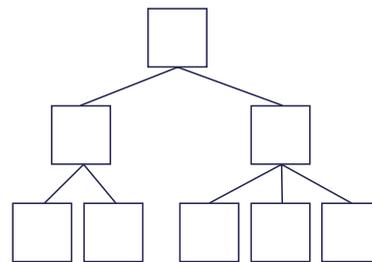


**Datenfluss:**  
Funktionen, Operational Activities, Interaktionen und Datenaustausch

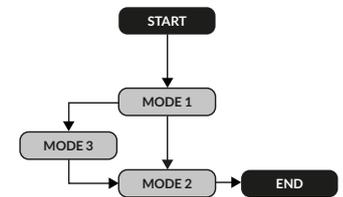
**Szenarien:**  
Actors, System, Komponenteninteraktionen und Datenaustausch



**Functional chains, Betriebliche Prozesse** durch Funktionen und Operational Activities

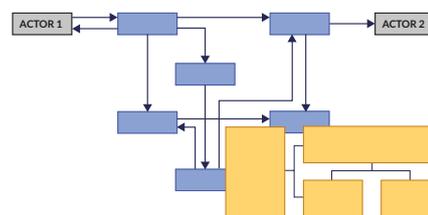
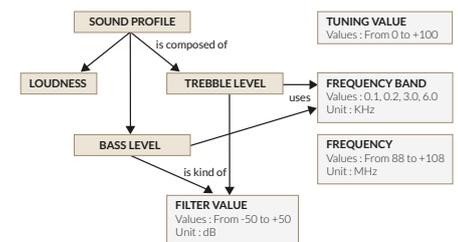


**Aufteilung von Funktionen & Komponenten**



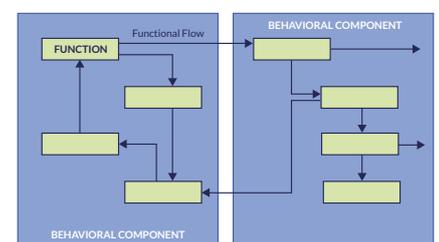
**Modes & States von Aktoren, System und Komponenten**

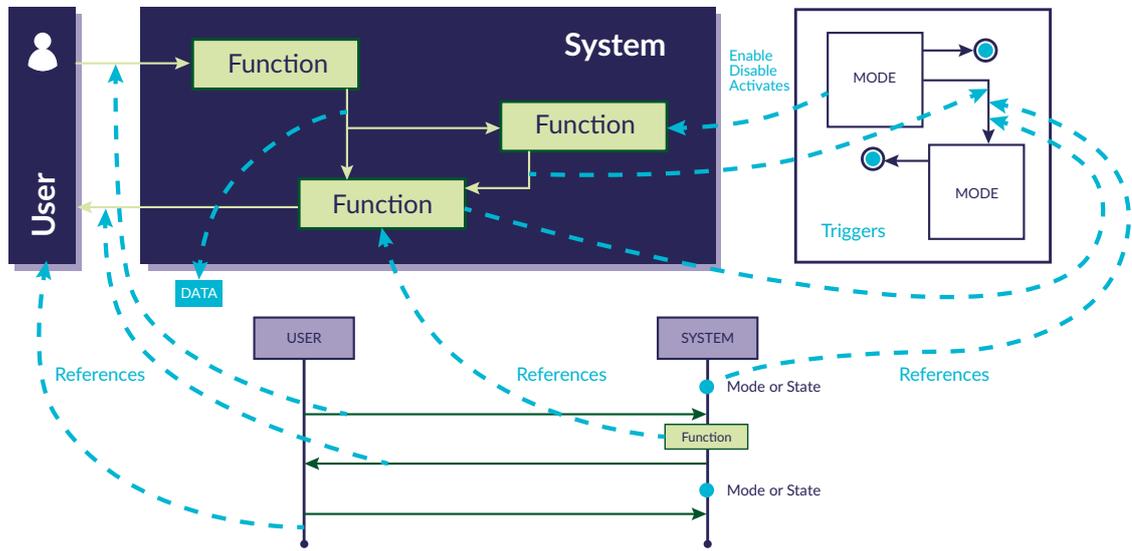
**Datenmodell: Datenfluss & Szenarien, Definition und Erläuterung der Schnittstellen**



**Komponentenverbindungen:**  
Alle Arten von Komponenten

**Zuweisung** der Operational Activities zu Aktoren, der Funktionen zu den Komponenten, der Verhaltenskomponenten zu den Implementierungskomponenten, der Datenflüsse zu den Schnittstellen, der Elemente zu den Konfigurationsobjekten





## Lösungen gegenüber nichtfunktionalen & industriellen Anforderungen prüfen und verifizieren

Methodische Schritte	Beispiele für Performance Anforderungen	Beispiele für Sicherheitsanforderungen
BETRIEBLICHE ANFORDERUNGSANALYSE	Max. Reaktionszeit auf Gefährdung	Ungewollte Ereignisse
FUNKTIONELLE/NICHTFUNKTIONELLE BEDARFSANALYSE	Functional Chain (FC) zur Reaktion auf Gefährdung. Maximale erlaubte Latenz auf der FC	Mit Ereignissen verbundene kritische Functional Chains
DESIGN DER LOGISCHEN ARCHITEKTUR	Komplexität der Verarbeitung und des Datenaustauschs Zuweisung der Functional Chains	Sichern der Functional Chains durch redundante Wege
DESIGN DER PHYSIKALSCHEN ARCHITEKTUR	Ressourcenverbrauch auf der FC Folgende Berechnungslatenz	Häufige Fehlermöglichkeiten Fehlerfortpflanzung auf der FC
ENTWICKLUNGSVEREINBARUNGEN UND IVVQ	Zugewiesene Ressourcen zur akzeptablen Latenz	Benötigter Zuverlässigkeitslevel

- ✓ Kosten und Planung
- ✓ Schnittstellen
- ✓ Leistung

- ✓ Wartbarkeit
- ✓ Betriebs-/Informationssicherheit
- ✓ ...

- ✓ IVVQ
- ✓ Produktpolitik

