**Beitrag P: Ulrich Meissen, Stefan Pfennigschmidt, Markus Hardt, Daniel Faust, Frank Fuchs-Kittowski**

# Eine Mikroservice-basierte Referenzarchitektur für interoperable, flexible und robuste Warnsysteme

# A Microservice Based Reference Architecture for Interoperable, Flexible and Robust Warning Systems

Ulrich Meissen[12], Stefan Pfennigschmidt[1], Markus Hardt[1], Daniel Faust[1], Frank Fuchs-Kittowski[12]

[1]*Fraunhofer FOKUS, {ulrich.meissen, stefan.pfennigschmidt, markus.hardt, daniel.faust, frank.fuchs-kittowski}@fokus.fraunhofer.de*
[2]*HTW Berlin, {ulrich.meissen, frank.fuchs-kittowski}@htw-berlin.de*



**Abstract**

Currently, in environmental systems, and disaster management in particular, we are witnessing a general shift from isolated monolithic implementations to highly robust and functionally distributed information processing in a systems of systems approach. Warning systems can serve as a representative example of a class of applications where these new requirements can be observed. Currently, warning systems are primarily specific developments with a generally similar but custom-made and often monolithic architecture. In order to meet the new challenges, appropriate reference architectures for these types of system classes are required to prevent a complete reimplementation of each new system. This paper presents a reference architecture for warning systems developed and tested in the basis of the KATWARN warning systems. The approach has been tested in a wide range of applications, ranging from mass application of a public warning system with approximately 4 million users in Germany and Austria to local target group-specific warning systems for critical infrastructures and large chemical plants. Based on the principles of event-based and asynchronous architecture paradigms, the implementation is described in a Microservice-based reference architecture. The architecture shows how challenging requirements in regards to performance, robustness and scalability can be met in such distributed process environments.

**Zusammenfassung**

Derzeit erleben wir in Umweltsystemen und insbesondere im Katastrophenmanagement einen generellen Wandel von isolierten monolithischen Implementierungen hin zu hochgradig robuster, verteilter Funktionalität und Informationsverarbeitungsketten in einem „Systems of Systems“-Ansatz. Warnsysteme können als repräsentatives Beispiel für eine Klasse von Anwendungen dienen, in denen diese neuen Anforderungen beobachtet werden können. Derzeit sind Warnsysteme hauptsächlich spezifische Entwicklungen mit einer im Allgemeinen ähnlichen, aber einzelangefertigten und oft monolithischen Architektur. Um den neuen Herausforderungen gerecht zu werden, sind entsprechende Referenzarchitekturen für diese Art von Systemklassen notwendig, um eine komplette Neukonzeption bei jeder neuen Implementierung zu verhindern. Dieser Beitrag stellt eine Referenzarchitektur für Warnsysteme vor, die auf der Grundlage mehrjähriger Forschung entwickelt und in den KATWARN-Warnsystemen getestet wurde. Der Ansatz wurde in einem breiten Anwendungsfeld getestet, von der Massenanwendung eines Bevölkerungswarnsystems mit ca. 4 Millionen Nutzern in Deutschland und Österreich bis hin zu lokalen zielgruppenspezifischen Warnsystemen für kritische Infrastrukturen und große Chemieanlagen. Basierend auf den Grundlagen von ereignisbasierten und asynchronen Architekturparadigmen wird die Implementierung in einer Mikroservice-basierten Referenzarchitektur beschrieben. Die Architektur zeigt, wie damit hohe Anforderungen an Performanz, Robustheit und Skalierbarkeit in verteilten Prozessumgebungen erfüllt werden.

# 1 Einleitung

Mit der stark wachsenden Verfügbarkeit von (Echtzeit-)Sensordaten und der nahezu ubiquitären Verbreitung von mobilen Geräten wurden im Bereich Monitoring und Dissemination im vergangen Jahrzehnt die entscheidenden Voraussetzungen für die Realisierung effektiver Warnsysteme für verschiedenste Einsatzbereiche und Zielgruppen geschaffen [Meissen 2012]. Diese reichen von generellen Bevölkerungswarnsystemen vor Naturgefahren wie Wetter-, Hochwasser und Katastrophenwarnungen für die Bevölkerung bis hin zu lokalen industriellen Anwendungen wie Chemieunfallwarnungen auf Werksgeländen.

Mit diesem breiten Einsatzspektrum und neuen technischen Anbindungsmöglichkeiten im Bereich Internet of Things (IoT) steigen sowohl die funktionalen aber auch insbesondere die nicht-funktionalen Anforderungen an die technische Realisierung derartiger Warnsysteme. Im funktionalen Bereich sind dies beispielsweise die situative

sowie zielgruppenspezifische Anpassung von Warnnachrichten und Verhaltenshinweisen oder die Erweiterung durch Crowdsensing-Elemente wie Tasking- und Rückmeldungsfunktionen [Meissen et al. 2017]. Im Bereich der nicht-funktionalen Anforderungen sind dies vornehmlich Robustheit, Performance und Skalierbarkeit (z.B. für Bevölkerungswarnsysteme mit Millionen von Nutzern) aber auch immer mehr Interoperabilität, Integrationsfähigkeit sowie funktionale Flexibilität in Richtung einer zunehmenden Einbettungs- und Vernetzungsnotwendigkeit mit verschiedensten Infrastrukturen und Systemen der Gefahrenabwehr.

Im letzten Jahrzehnt wurden zahlreiche Warnsysteme als Einzelentwicklungen mit verschiedensten Architekturen immer wieder neu entwickelt, da es derzeit keine praktisch anwendbaren Referenzarchitekturen für diese Klasse von IT-Systemen gibt. Erfahrungsgemäß führt dies nicht nur zu einem höheren Entwicklungsaufwand, sondern zu – in diesem kritischen Anwendungsfeld besonders problematischen – Defiziten im Bereich der nicht-funktionalen Qualitätseigenschaften wie Performance, Robustheit und Skalierbarkeit, die in der Regel nur durch aufwendige Re-Engineering-Maßnahmen behoben werden können. Zusätzlich zeigen bisherige Systeme mit einem speziellen Einsatzzweck Defizite im Bereich der Interoperabilität und funktionalen Flexibilität, die zunehmend an Bedeutung gewinnen.

Der vorliegende Beitrag stellt eine Referenzarchitektur vor, die auf Basis langjähriger Forschungsarbeiten entwickelt und in dem Warnsystem KATWARN [CombiRisk 2018] in verschiedensten Einsatzkontexten von der Massenwendung eines Bevölkerungswarnsystem mit über 4 Millionen Nutzern in Deutschland und Österreich bis hin zu lokalen zielgruppenspezifischen Warnsystemen für kritische Infrastrukturen und Chemiebetriebe erprobt wurde. Ausgehend von den Grundlagen event-basierter und asynchroner Architekurparadigmen wird deren Umsetzung in einer Mikroservice-basierten Referenzarchitektur beschrieben. Anhand der Architektur wird aufgezeigt, wie damit hohe Performance-, Robustheits- und Skalierbarkeitsanforderungen erfüllt werden können. Zusätzlich beschreiben wir die zentralen Interoperabilitäts- und Erweiterbarkeitseigenschaften durch einfache Service-Verkettungs- und Topologie-Anpassungsmaßnahmen, durch die beispielsweise eine einfache Realisierung einer funktionalen Erweiterung zu einem Crowdsensing- und Helfersystem ermöglicht wird.

Der Beitrag ist wie folgt aufgebaut: In Kapitel 2 werden bisherige Ansätze im Bereich der Architekturen von Warnsystemen beschrieben. Darauf folgend werden in Kapitel 3 auf Basis von einer Klassifikation von Anwendungsszenarien für Warnsysteme die Kriterien im Bereich der nicht-funktionalen Anforderungen aufgestellt, die eine Referenzarchitektur für Warnsysteme erfüllen sollte. In Kapitel 4 wird die grundlegende Referenzarchitektur und deren praktische Umsetzungen in den Systemen KATWARN und KATRETTER beschrieben. Darauf aufbauend erfolgt in Kapitel 5 eine Evaluierung anhand der in Kapitel 3 aufgestellten Kriterien. Abschließend werden in Kapitel 6 die Ergebnisse zusammengefasst und künftige Nutzungspotentiale der Referenzarchitektur aufgezeigt.

## 2 Existierende Ansätze für Warnsystemarchitekturen

Das hier betrachtete Anwendungsfeld ist ein Teilgebiet von Frühwarnsystemen Vereinfachend umfassen die funktionalen Komponenten einer vollständigen Frühwarnsystemimplementierung Monitoring, Entscheidungsunterstützung (d. H. Gefahrenerkennung, Risikobewertung) und reine Warnsysteme. Die Ziele eines Monitoring-Systems sind die laufende Überwachung gegebener Indikatoren durch Messungen (üblicherweise durch physikalische Sensoren, virtuelle Sensoren oder Sensorsysteme) bzw. Schätzungen in einer gegebenen Frequenz, um die gewonnenen Daten bereitzustellen. Ziel des Gefahrenerkennungssystems ist es, Gefährdungsgefahren zu erkennen und die daraus resultierenden Gefahrensituationen abzuschätzen, die beide auf der Analyse verfügbarer Beobachtungsinformationen beruhen. Ziel eines Risikobewertungssystems ist es, Risiken im Zusammenhang mit einer bestimmten Gefährdungslage zu bewerten. Schließlich ist das eigentliche Warnsystem für die Generierung von gezielten Warnungen aus den erkannten Gefahren- und Risikoinformationen sowie für deren effiziente Verbreitung verantwortlich.

Im Folgenden konzentrieren wir uns auf die Gruppe der Warnsysteme, die auch als Alarmierungssysteme bezeichnet werden. Wir sollten jedoch den allgemeinen Kontext von Früharnsystemen berücksichtigen, da Warn- oder Alarmietrungssysteme immer Teil des gesamten Frühwarnsystems sind. Dies wird insbesondere wichtig, wenn wir die nicht-funktionalen Anforderungen wie Interoperabilität oder Flexibilität diskutieren. Zum Beispiel muss in einem Frühwarnsystem für mehrere Gefahrenarten (multi-

hazard) ein Warnsystem an mehrere Überwachungs-, Gefahrenerkennungs- und Risikobewertungssysteme anschließbar sein.

Mit den Möglichkeiten neuerer Informationstechnologien und -infrastrukturen hinsichtlich Sensorik, Rechenleistung, durchgängiger Vernetzung sowie Informationsverbreitung erleben wir seit den frühen 2000er Jahren eine Entwicklung von neuen Frühwarnsystemen vor allem im Bereich von Naturgefahren wie Extremwetter, Erdbeben oder Tsunamis. Ein erster Versuch, die Architektur von EWS allgemein zu strukturieren und wiederverwendbare Komponenten bereitzustellen, wurde im Rahmen des Projekts ASGARD zwischen 2002 und 2004 von der Joint Reseerch Center durchgeführt [Jacobson 2004]. Mit der Einführung des Common Alerting Protocol (CAP) im Jahr 2006 [Botterell 2006] wurde ein wichtiger Meilenstein in Bezug auf die notwendige Interoperabilität für Warnsysteme erreicht. Im folgenden Jahrzehnt wurde CAP zur "lingua franca" für den Austausch von Warnungen zwischen und sogar innerhalb von Warnsystemen. In Bezug auf Standardisierung, Best Practices und Referenzarchitekturen gibt es einige Arbeiten, beispielsweise auf dem Gebiet der Sensordatenverarbeitung [Botts et al. 2008], der Entscheidungsunterstützung [Balis et al. 2011, Babitski et al. 2011] sowie der Warnung und Alarmierung. Für allgemeine Warnsysteme wurde in [Meissen & Voisard 2010] ein Vorschlag für eine Referenzarchitektur vorgestellt.

Wie jedoch in einer Analyse von bestehenden Frühwarnsysteme [Moßgraber 2017] gezeigt wurde, handelt es sich bei den derzeitigen Implementierung hauptsächlich um spezifische Entwicklungen mit einer im Allgemeinen ähnlichen, aber spezifisch entwickelten Architektur. Trotz der bestehenden vorgeschlagenen Referenzarchitekturen für Frühwarnsysteme scheint der Einfluss solcher Ansätze auf aktuelle Implementierungen eher begrenzt zu sein. Nach derzeitigem Kenntnisstand, hat keiner der in der Literatur voregstelleten Ansätze ihre Tauglichkeit über einen Prototyp oder Pilot hinaus in einer realen Massenanwendung bewiesen.

Eine der Ursachen ist sicherlich, dass die bestehenden Referenzarchitekturvorschläge zwar eine durchdachte funktionale Trennung und Komponentendefinition vorsahen, aber weniger Aspekte der Implementierungs- und Betriebsebene berücksichtigten. Auch wenn eine Referenzarchitektur weitgehend umsetzungsunabhängig sein sollte, ist es in der Praxis entscheidend, dass Aspekte der Implementierung berücksichtigt

werden. Insbesondere bei der Kombination neuer architektonischer Paradigmen, wie etwa Mikroservices, haben diese Aspekte einen starken Einfluss auf die Ausarbeitung einer effektiven und tatsächlich nutzbaren Referenzarchitektur mit signifikanten Vorteilen im Hinblick auf nicht-funktionale Anforderungen.

In unseren Forschungsarbeiten, die mit Referenzarchitekturen auf Basis klassischer, datenbankzentrierter, schichten- und serviceorientierter Architekturen begannen, mussten wir feststellen, dass derartige Ansätze ihre Grenzen nicht so sehr in funktionalen, sondern vor allem in nicht-funktionalen Anforderungen haben, insbesondere wenn es zu Leistungs- und Skalierbarkeitsanforderungen in Massenanwendungen kommt (z. B. zeitkritische Warnungen für mehr als 1 Mill. Benutzer). Daher haben wir in den letzten fünf Jahren unsere bestehende Referenzarchitektur für die Warnsysteme WIND [Meissen et al. 2013], SAFE [Klafft et al. 2009] und KATWARN grundsätzliche überarbeitet und in neuen Versionen dieser Systeme implementiert. Die neue Referenzarchitektur und die Ergebnisse der praktischen Evaluierung für verschiedene Anwendungskontexte werden in dieser Arbeit vorgestellt.

## 3 Nicht-funktionale Anforderungen

Wie in Kapitel 2 dargestellt, zielt unsere Referenzarchitektur auf die nicht-funktionalen Aspekte eines Warnsystems ab. Um unseren Ansatz zu bewerten, definieren wir eine Reihe von nicht-funktionalen Anforderungen die im Anwendungskonetxt Frühwransysteme von besonderer Bedutung sind. Dabei teilen wir die nicht-funktionalen Anforderungen in den folgenden drei Kategorien auf:

(1) Funktionsqualitäten,

(2) Betriebsqualitäten und

(3) Entwicklungsqualitäten.

Tabelle 1 gibt einen Überblick über die berücksichtigten Anforderungskriterien:

| Kategorie | Anforderung |
|---|---|
| **Funktionsqualitäten** | **Compliance** |
| | **Fehlertoleranz** |
| | **Testbarkeit** |
| | **Rückverfolgbarbarkeit** |
| **Betriebsqualitäten** | **Leistung** |
| | **Verfügbarkeit** |
| | **Zuverlässigkeit** |
| | **Robustheit** |
| | **Sicherheit** |
| **Entwicklungsqualitäten** | **Portabilität** |
| | **Modifizierbarkeit** |
| | **Interoperabilität** |
| | **Skalierbarkeit** |

Tabelle 1: Kategorisierung nicht-funktionaler Anforderungen für Warnsysteme [Meissen 2012]

Unter den funktionalen Qualitäten listen wir die Qualitätsparameter der vom System bereitgestellten Funktionen auf. Ein entscheidender Faktor ist die Kritikalität des Systems. Warnsysteme sind im Allgemeinen kritische Systeme. Ihre Funktion muss von höchster Qualität sein, da jede Störung ernsthafte Auswirkungen auf das langfristige Vertrauen der Betroffenenm haben kann (z.B. wenn ein Fehlalarm ausgelöst wird bzw. im Katastrophenfall keine Warnung erfolgt). Diese Kritikalität gilt besonders für Bevölkerungswarnsysteme. In diesem Zusammenhang sind die wichtigsten nicht-funktionalen Anforderungen im Bereich der Funktionsqualitäten, Compliance, Fehlertoleranz, Testbarkeit und Rückverfolgbarkeit.

Unter den Betriebqualitäten beschreiben wir die für den Betrieb des Systems relevanten Eigenschaften. Im Kontext von Frühwarnsystemen und deren Krikialität sind wiederum die wichtigsten nicht-funktionalen Anforderungen an die Betriebsqualitäten Leistung, Zuverlässigkeit, Verfügbarkeit, Robustheit und Sicherheit.

Unter den Entwicklungs oder Evolutionsqualitäten beschreiben wir die Eigenschaften des Systems für die langfristige Anpassung an sich ändernde Anforderungen. Ausschlaggebend sind die Heterogenität der Anwendungsszenarien, die erwartete Dynamik der Anforderungen und die postulierte Kosteneffizienz. Frühwarnsysteme arbeiten im Allgemeinen in dynamischen Umgebungen, in denen sich Erkennungs-,

Vorhersage- und Warntechnologien sowie Reaktionsstrategien im Laufe der Zeit ändern können. Ein wichtiger Aspekt ist, dass Frühwarnsysteme in der Regel für einen langfristigen Betrieb geplant sind, da die mit dem System adressierten Gefahren oft eine permanente Bedrohung darstellen. Daraus folgen die hohehn Anforderungen, die an die langfristige Entwicklungsqualitäten von Frühwarnsystemen gestellt werden. Die relevanten nicht-funktionalen Anforderungen in dieser Kategorie sind Portabilität, Modifizierbarkeit, Interoperabilität und Skalierbarkeit.

In dieser Arbeit konzentrieren wir uns auf die nicht-funktionale Anforderungspaare, die bei der Entwicklung effizienter Warnsysteme für die Praxis entscheidend sind. Diese sind Leistung/Skalierbarkeit, Zuverlässigkeit/Fehlertoleranz, Portabilität/ Modifizierbarkeit und Rückverfolgbarkeit/Testbarkeit.

## 4 Architektur und Implementierung

In diesem Kapitel beschreiben wir eine ereignis- und k-basierte Architektur für Warnsysteme. Der ereignisgesteuerte Ansatz bedeutet vereinfacht, dass die Interaktion zwischen Komponenten in dem System ausschließlich oder hauptsächlich durch interne oder externe Ereignisse ausgelöst wird und die Architektur daruf optmiert wird. Ein Frühwarnsystem kann allgemein als ein System charakterisiert werden, das durch externe Ereignisse (z.B. neue Beobachtungen) ausgelöst wird, diese Ereignisse verarbeitet (z.B. Erkennung einer Gefahr in den Daten) und die resultierenden Warnungen als Ereignisse für Nutzer bzw. angeschlossene Systeme weiterleitet. Daher ist es sinnvoll, alle Komponenten auf dem kritischen Pfad eines Frühwarnsystems basierend auf ereignisgesteuerten Prinzipien zu entwerfen. Soweit sind wir konform zur bisherigen Referenzarchitektur, wie sie in [Meissen & Voisard 2010] beschrieben wurde.

Der Hauptunterschied besteht nun im Schritt vom serviceorientierten, datenbankzentrierten und schichtenbasierten Ansatz hin zu einem rein Mikroservice-basierten Ansatz, der auch im Kontext einer ereignisgesteuerten Architektur sinnvoll ist. Auch wenn Mikroservices als eine konsequente Weiterentwicklung oder eine Variante von Service-orientierten Architekturen (SOA) gesehen werden kann, hat eine Ablösung von SOA entscheidende Auswirkungen auf die Architektur. In einer vereinfachten Beschreibung besteht ein Mikroservice-basiertes System aus einer

Anzahl funktionaler kleiner, loser gekoppelter und in sich geschlossener Services, die Implementierungsdetails verbergen und über leichtgewichtige Protokolle über feinspezifizierte Schnittstellen kommunizieren. Die Services sollten normalerweise ihren eigenen Datenspeicher haben und autonom betrieben werden können. Die Hauptaufgabe beim Entwurf von Mikroservice-basierten Architekturen ist die funktionale Dekomposition und das Schnittstellenspezifikation der einzelen Mikroservices. Die Vor- und Nachteile dieses Ansatzes im Kontext von Warnsystemen werden in Kapitel 5 diskutiert. Hier konzentrieren wir uns auf die Beschreibung der Architektur, die in Abbildung 1 dargestellt ist.

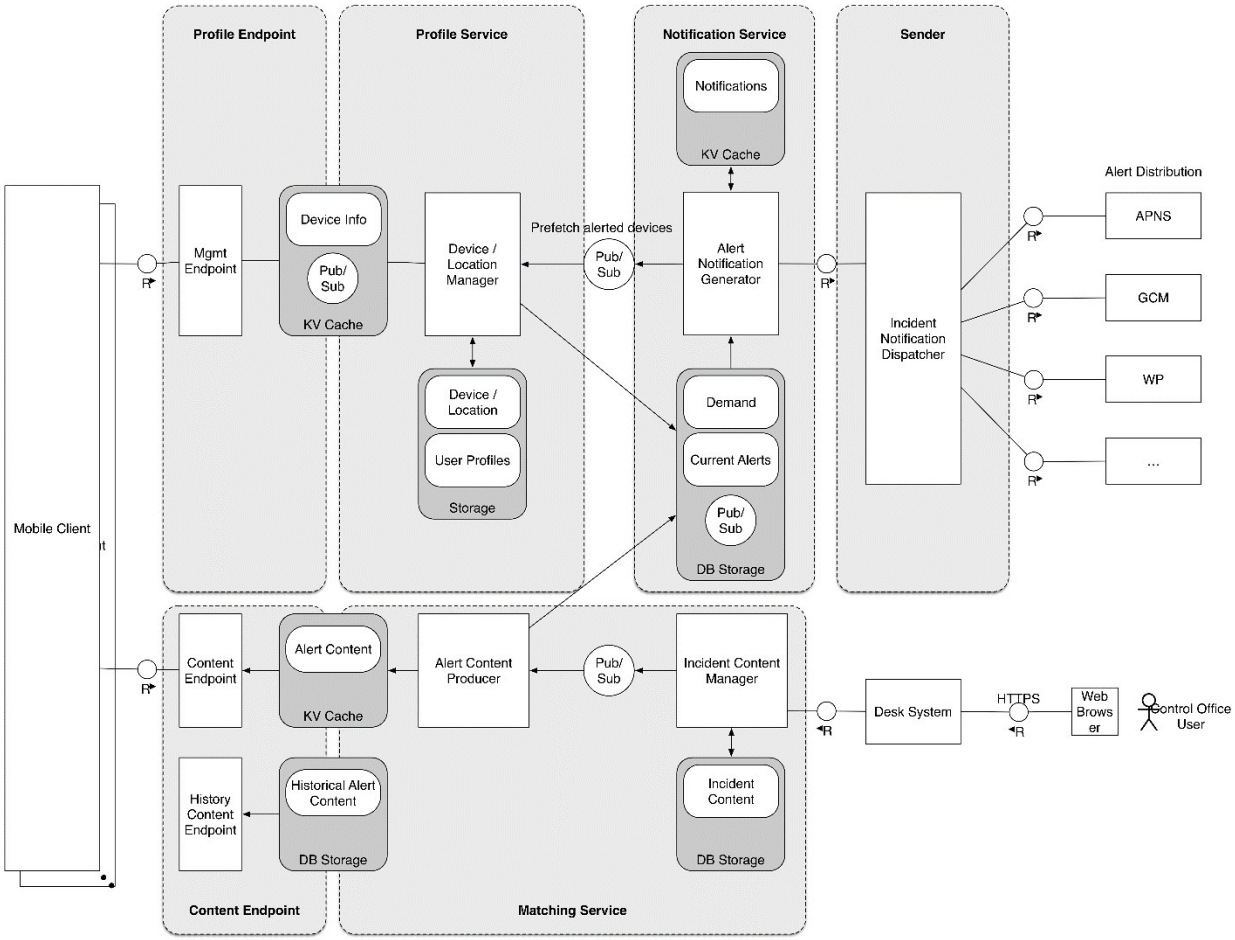


Abbildung 1: Architektur für Warnsysteme

Die Hauptkomponenten der Architektur sind für die folgenden Funktionen verantwortlich:

**Verarbeitung von externen Ereignissen über den *Matching Service***: Der Das *Incident Content Management* ist für die Verwaltung eingehender Ereignisse

verantwortlich. Es nimmt Ereignismeldungen über verschiedene externe Schnittstellen entgegen und speichert diese persistent im *Incident Content Storage*. Basierend auf einem Abgleich des aktuellen Bedarfs und existierenden Warnungen werden neue oder aktualisierte Ereignisse als Vorfälle an den *Alert Notification Generator* und damit an den *Notification Service* weitergeleitet.

**Relevante Ereignisse über den *Notification Service* bearbeiten**: Der *Alert Notification Generator* ist verantwortlich für das Matching von aktuellen Endnutzerbedarfen, die in Abonenements gespeicherte werden (z.B. aktuelle Position) und Ereignissen. Der Service hat Zugriff auf den *Demand-/Alert-Storage*, eine spezielle Laufzeitdatenbank, in der die Abonnement- und Ereignisdaten bereits kombiniert wurden. Die Hauptaufgabe des Diesntes besteht dann darin, die Adressen der relevanten Geräte der Endnutzer abzurufen und diese Daten an den *Alert Notification Dispatcher* zu senden. Darüber hinaus verfügt diese Komponente über einen Speicher, in dem der Benachrichtigungsstatus der einzelnen Geräte gespeichert ist. Der *Alert Notification Generator* informiert außerdem andere Services darüber, von welchen Geräten erwartet wird, dass sie demnächst Abfragen senden, bevor die Zustellung der Warnung ausgelöst wird. Dies macht es möglich, das System so optimieren, dass die angeforderte Information über einen Cache vorab zur Verfügung gestelt werden können, was die Leistungsfähigkeit bei Abfargen erheblich erhöht.

**Warnmeldungen durch den *Sender* verbreiten (Push)**: Der *Alert Notfication Dispatcher* ist für das Versenden der Benachrichtigungen über verschiedene Kanäle verantwortlich. Er empfängt die vom *Alert Notification Generator* erzeugten Warnungen zusammen mit den entsprechenden Geräteadressen und sendet die Benachrichtigungen.

**Warninformationen durch den *Content-Endpoint* abrufen (Pull)**: Dieser Service bietet eine Schnittstelle, über die Inhalte abgefragt werden können. Dies beinhaltet Abfragen von ausgegebenen Warnunge und relevanten Zusatzinformationen. Die Daten stammen aus dem *Alert Content Cache*. Der *Content-Endpoint* ist auf maximale Skalierbarkeit ausgelegt und verwendet dynamische Lastverteilung. Der *Alert Content Cache* dient als Puffer für den zu liefernden Inhalt (z. B. Zusatztexte und Warnkarten).

**Verwalten von persönlichen und gerätespezifischen Informationen über den *Profile-Endpoint* und *Profile Service***: Der *Profile-Endpoint* stellt die Schnittstelle zur

Verfügung, über die der Benutzer Profilinformationen abfragen, hinzufügen oder ändern kann. Ähnlich wie der *Content-Endpoint* verwendet der *Profile-Endpoint* eine dynamische Lastverteilung, um unter anderem mehrere Profilaktualisierungen in kurzer Zeit zu ermöglichen. Der *Profile-Endpoint* ruft Antworten auf Abfragen aus dem *Profie-Info-Cache* ab, der häufig angeforderte Daten sowie zuletzt geänderte Profilinformationen in einem leistungsstarken Schlüssel/Wert-Cache enthält. Der *Device/Location-Manager* ist für die Verwaltung aller profilbezogenen Daten im System verantwortlich. Nutzeranfragen werden an den Manager weitergeleitet, wenn sie nicht direkt vom *Profile-Info-Cache* beantwortet werden können.

Die auf dieser Architektur basierenden Warnsysteme wurden unter Verwendung von serverseitiger JavaScript-Technologie einschließlich über die in dem MEAN-Stack vorgesehenen Technologien wie MongoDB, Express.js, AngularJS, Node.js sowie Redis als Schlüsselwertspeicher implementiert.

# 5 Evaluierung

Methoden zur Evaluierung von Architekturen stell schon für sich ein eigenes Forschungsgebiet dar. Nach Kenntnisstand gibt es keine formale Bewertungsmethode für nicht-funktionale Anforderungen auf theoretischer Ebene. Tatsächlich hängt die Bewertungsmethode von den zu betrachtenden Kriterien ab und kann oft nur in einer Kombination von empirischen und logischen Schlussfolgerungen bewertet werden.

In Abschnitt 3 haben wir relevante nicht-funktionale Kriterien zur Bewertung von Warnsystemen vorgestellt. In dieser Arbeit konzentrieren wir uns bei der Evaluierung auf die relevantesten Kriterienpaare, nämlich Leistung/Skalierbarkeit, Zuverlässigkeit/Fehlertoleranz, Portabilität/Modifizierbarkeit und Rückverfolgbarkeit/ Testbarkeit. Im Folgenden betrachten wir die Eigenschaften der vorgestellent Architektur einzeln nach diesen Kriterien:

## 5.1 Leistung und Skalierbarkeit

Ein öffentliches Warnsystem wie KATWARN mit derzeit über 4 Millionen abonnierten Endnutzern und der Anforderung, innerhalb weniger Minuten mehrere hunderttausend Endnutzer individuell alarmieren zu können, ist offensichtlich ein adäquates empirisches Testfeld für die Leistungsfähigkeit und Skalierbarkeit von Warnsystemen. Eine wesentliche Motivation für ein Redesign der bisherigen SOA-basierten, zentralen

datenbankorientierten und schichten-orientierten Architektur von KATWARN waren die Grenzen, die bei mehr als 1 Million Endnutzern erreicht wurden. Aufgrund von effizienten Caching-Mechanismen basierend auf einfachen Schlüssel-Wert-Speichern, der strikten Einhaltung des Prinzips des *Seperation of Concerns* bei der Service-Abgrenzung, der optimierten, flexiblen und verteilbaren Ereignisverarbeitung in einen Mikroservice-basierten Ansatz ist die neue Architektur weit überlegen zu der früheren Architektur in Bezug auf die Leistung (teilweise um den Faktor 12). Selbst wenn alle Optimierungspotentiale der früheren Architektur genutzt worden wären, hätte die durch die Architektur gegebenen strukturellen Einschränkungen kein wesentlich besseres Ergebnis erlaubt. Angesichts der aktuellen KATWARN-Produktionsinfrastruktur von 14 Servern (Intel® CoreTM i7-6700 Quad-Core Skylake 32 GB DDR4 RAM, 2 x 250 GB SATA 6 GB / s SSD (RAID 1), 1 GBit / s-Port) kann leicht berechnet werden, was ein beispielweise ein Leistungssteigerungsfaktor von12 in Bezug auf Hardware und Wartungskosten bedeutet.

Draüberhinaus bietet die neue Architektur viel gezieltere und effizientere Möglichkeiten der Skalierbarkeit. Während die alte, monolithische und auf zentrale Datenbanken orientierte Architektur nur die Möglichkeit bot, über logische Lastverteilung und neue Instanzen des gesamten Warnsystem zu skalieren, kann man sich bei der neuen Architektur die spezifische Skalierung einzelner Services konzentrieren und diese sogar dynamisch durchführen, indem zusätzliche Services zur Laufzeit gestartet werden. Beispielsweise besteht die höchste Verarbeitungslast bei den Services *Sender* und der *Content-Endpoint*. Daher werden diese Services derzeit als 8 identische Instanzen auf einzelnen Hochleistungsservern betrieben, um ausreichend Leistungsspielraum für bis zu 10 Millionen Endnutzer bereitzustellen.

## 5.2 Zuverlässigkiet und Fehlertoleranz

In diesem Bereich müssen wir zwischen zwei Aspekten unterscheiden: erstens die Fähigkeit, Fehler zu vermeiden, und zweitens die Robustheit des Systems, wenn Fehler auftreten. Bei der Entwicklung kritischer Systeme muss oft ein Kompromiss zwischen beiden Aspekten gefunden werden. Ein Hauptkritikpunkt bei der Verwendung von Mikroservices im Kontext kritischer Systeme ist die lose Kopplung zwischen Services durch hauptsächlich asynchrone Kommunikation, die auch als "Fire and Forget"-Paradigma bezeichnet wird. Eines der wichtigsten Mikroservics-Prinzipien besteht darin, dass der Ausfall eines Services die Reaktion auf oder Auslösung von

Ereignissen in anderen Sevices nicht blockieren sollte. Wenn ein Service fehlschlägt, kann er leicht (sogar automatisch) neu gestartet werden. Asynchrone Ereignisse von oder zu diesem Service können jedoch in dieser Zeit verloren gehen. Offensichtlich ist ein solches Verhalten hinsichtlich der Robustheit vorteilhaft, erzeugt jedoch einen möglichen Anteil von Funktionsfehlern. (in unserem Fall Warnmeldung, die nicht an die Endbenutzer geliefert wird). Es können jedoch Maßnahmen gegen solche Funktionsfehler in einem Mikroservice-basierten System ergriffen werden, entweder durch Einführung von Zuständen in den Services (was gegen das allgemeine Prinzip von Mikroservices verstoßen würde, die im Allgemeinen zustandslos solletn) oder durch Verwendung von Transaction-Caches zwischen kritischen Services.

Der große Vorteil unserer neuen Architektur ist hier die erhöhte Robustheit, insbesondere im Kontext einer Massenanwendungen. Während die alte Architektur eine teure, vollständige Systemredundanz erforderte, z. B. falls eine Software- oder Hardwarekomponente ausfällt, weist die neue Architektur selbst dann eine hohe Robustheit auf, wenn ein kompletter Server ausgefallen ist, da identische Services auf anderen Servern sofort für den fehlgeschlagenen Service einspringen können. Unsere Messungen zeigen, dass dabei die Kosten von Funktionsfehlern relativ gering sind (unter 0,00003% der möglicherweise nicht ausgelieferten Einzelwarnungen).

### 5.3 Portabilität und Modifizierbarkeit

Wie in vielen anderen Bereichen beobachten wir im Katastrophenmanagement einen generellen Wandel von isolierten monolithischen Implementierungen hin zu vernetzten Aufgaben und Prozessen in einem "*Systems of Systems*"-Ansatz. Dies erhöht nicht nur die Anforderungen an die Interoperabilität, sondern auch die funktionale Anpassungsfähigkeit von Systemen. Darüber hinaus sollten die Systeme für verschiedene Aufgaben, die sich im Katastrophenmanagement entwickeln, immer flexibler und erweiterbarer werden. In [Meissen et al. 2017] haben wir bereits die funktionelle Flexibilität der Architektur gezeigt, in der das Warnsystem KATWARN durch Integration von Crowd-Sensing-Elementen zu einem Helfersystem erweitert wird. In dieser Arbeit zeigen wir die Flexibilität der Architektur, in dem eine neue Funktionalität bereitgestellt werden kann, indem nur die Topologie der Architektur in einer verteilten Verarbeitungsinfrastruktur geändert wird:

Im Jahr 2017 wurde KATWARN als nationales Bevölkerungswarnsystem für Österreich eingeführt. Eine wesentliche Herasuforderung für die Implementierung war die Bereitstellung einer unabhängigen Warnsystem-Infrastruktur für Österreich und gelichzeitig eine "Roaming" -Funktionalität zwischen dem deutschen und dem österreichischen System (d.h. ein deutscher Nutzer erhält Warnungen in Österreich und umgekehrt, ohne sich eine neu App laden zu müssen). Die Abbildungen 3 und 4 zeigen die topologische Änderung der Architektur mit der die Anforderung ohne Änderungen der Services realisiert werden konnte.

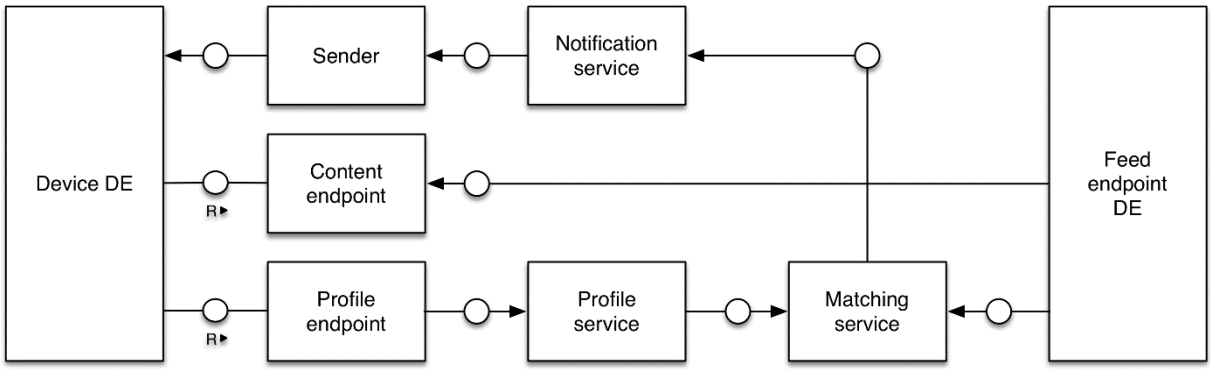


Abbildung 3: Vereinfachte Architektur des deutschen KATWARN-Systems

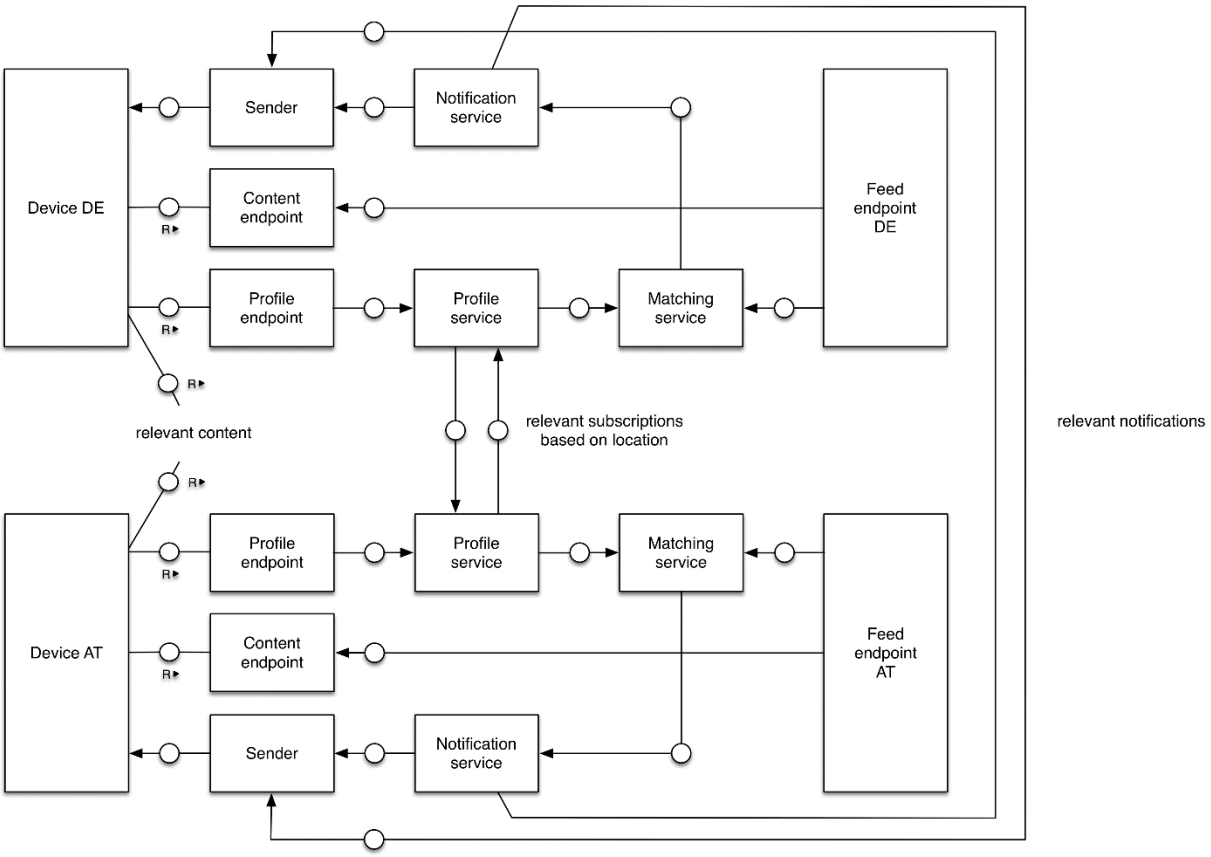


Abbildung 4: Architektur des Warnsystem für Deutschland und Österreich

In Abbildung 3 sehen wir eine vereinfachte Ansicht der Architektur es bisherigen Warnsystems für Deutschland. In Abbildung 4 sehen wir die implementierte Lösung für Österreich und Deutschland. Es werden nicht vorhergesehene Anforderungen erfüllt, indem lediglich relevante Dienste für die Roamingfunktionalität miteinander verbunden werden, ohne Schnittstellen zu ändern, Services hinzuzufügen oder den Code zu ändern.

Selbst wenn dieser Fall eher ein ideale Ausnahme darstellt, unterstreicht er doch stark die grundsätzlichen Qualitäten dieser Architektur in Bezug auf Portabilität und Modifizierbarkeit. In einem weiteren Schritt zielen wir auf eine Lösung ab, bei der österreichische und deutsche Services als Backup für Ausfälle in den Infrastrukturen eines Systems dienen können, wiederum allein durch Änderung der Topologie der Architektur.

### 5.4 Rückverfolgbarkeit und Testbarkeit

Einer der Hauptnachteile des neuen Architekturparadigmas ist die kompliziertere Rückverfolgbarkeit und Testbarkeit. Während das Verhalten eines einzelnen Services noch leicht verfolgt und getestet werden kann, ist das Verhalten der gesamten Service-Choreographie in dem integrierten System aufgrund der größeren Anzahl, der losen Kopplung und des asynchronen Verhaltens der Service viel komplexer als beispeilsweise in SOA. Das mögliche Hinzufügen gemeinsamer Transaktions-IDs für alle Dienste ist schwierig zu implementieren, würde die Kommunikation unnötig erhöhen und den Grundsatz der Unabhängigkeit der Dienste verletzen. Dies führt zu einem allgemeinen Problem der Verwaltung großer komplexer Mikroservice-basierter Architekturen. Aufgrund des relativ neuen Ansatzes sind für diese Probleme nach unserer Kenntnis noch keine geeigneten Managament-Werkzeuge entwickelt worden.

## 6 Zusammenfassung

In dieser Arbeit haben wir eine neue Referenzarchitektur für Warnsysteme basierend auf ereignisgesteuerten und Mikroservice-orientierten Paradigmen vorgestellt. Die Machbarkeit und die Vorteile des Ansatzes wurden anhand zentraler nicht-funktionaler Anforderungen aufgezeigt, die durch die aktuellen Implementierungen der KATWARN-Warnsysteme parktisch getestet wurden. Der Ansatz hat seine Hauptvorteile in Bezug auf Leistung/Skalierbarkeit, Zuverlässigkeit/Fehlertoleranz und

Portabilität/Modifizierbarkeit gegenüber klassischen SOA-Architekturen. Wir haben jedoch auch Schwachstellen in Bezug auf Rückverfolgbarkeit/Testbarkeit entdeckt, die mit einem allgemeinen Problem der Planung, Verwaltung und Steuerung großer und komplexer Mikroservice-basierter Systeme einhergehen. Es ist jedoch sehr wahrscheinlich, dass diese Probleme durch die Entwicklung neuer Tools im Rahmen immer breiteren Einsatzes von Mikroservice-basierter Systemen behoben werden.

## 7 Literaturverzeichnis


Babitski, G.; Bergweiler S.; Grebner O.; Oberle D.; Paulheim, H.; Probst F. (2011): *SoKNOS–using semantic technologies in disaster management software*. In Extended Semantic Web Conference (pp. 183-197). Springer, Berlin, Heidelberg.

Balis, B.; Kasztelnik M.; Bubak, M.; Bartynski T.; Gubała T.; Nowakowski P. and Broekhuijsen J. (2011): *The UrbanFlood Common Information Space for Early Warning Systems*. In Procedia Computer Science, Vol. 4, pp. 96-105, ISSN 1877-0509.

Botterell, A. (2006): *The Common Alerting Protocol: an open standard for alerting, warning and notification*. In Proceedings of the 3rd International ISCRAM Conference, pp. 497-503, Newark, NJ, USA.

Botts, M.; Percivall G.; Reed C.; Davidson J.(2008): OGC (R) *Sensor web enablement: Overview and high level architecture*. In Lecture Notes In Computer Science 4540, pp. 175–190.

CombiRisk GmbH (2018): *KATWARN-Website*, http://www.katwarn.de (aufgerufen am 5.5.2018).

Jacobson, M. (2004:) *Asgard System Description*. Directorate-General, Joint Research Centre (JRC), European Commission.

Klafft M.; Kräntzer T.; Meissen U.; Voisard, A. (2009): *Early warning systems in practice: Performance of the SAFE system in the field*. In Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, pp. 436-439.

Moßgraber, J. (2017): *Ein Rahmenwerk für die Architektur von Frühwarnsystemen, In Karlsruher Schriften zur Anthropomatik*, Band 29, ISBN 978-3—7315-0638-6.

Meissen, U. (2012): *Targeted Alerting in Early Warning Systems*, Freie Universität Berlin.

Meissen, U.; Fuchs-Kittowski, F.; Jendreck, M.; Pfennigschmidt, S.; Hardt, M.; Voisard, A. (2017): *A general system architecture and design for the coordination of volunteers for agile disaster response*. In: Proc. of the 14th International Conference on Information Systems for Crisis Response And Management (ISCRAM2017), pp. 890-900, ISCRAM.org, Albi, Frankreich, ISSN 2411-3387.

Meissen U.; Voisard A. (2010): *Towards a Reference Architecture for Early Warning Systems*. In Proceedings of International Conference on Intelligent Networking and Col-laborative Systems, IEEE, DOI: 10.1109/INCOS.2010.81.

Meissen, U.; Faust D.; Fuchs-Kittowski F. (2013): *WIND - A meteorological early warning system and its extensions towards mobile devices*. In Proceedings of the 27. Conference on Environmental Informatics, pp. 612-621, Shaker, Aachen, ISBN 978-3-8440-1676-5, ISSN 1616-0886.