

**Technisches Konzept**

**Projektname: FISI-Wetterstation**

**Version:** 0.1

**Status:** Entwurf

**Datum:**

**Autor:**

Inhaltsverzeichnis

[Inhaltsverzeichnis 2](#_Toc485133015)

[1. Einführung und Ziele / Fachliche Anforderungen 3](#_Toc485133016)

[1.1 Aufgabenstellung 3](#_Toc485133017)

[1.2 Qualitätsziele 3](#_Toc485133018)

[1.3 Stakeholder 3](#_Toc485133019)

[2. Randbedingungen 4](#_Toc485133020)

[2.1 Technische Randbedingungen 4](#_Toc485133021)

[3. Kontext 5](#_Toc485133022)

[3.1 Externe Schnittstellen 6](#_Toc485133023)

[3.1.1 Externe Schnittstelle 1 6](#_Toc485133024)

[4. Lösungsstrategie 7](#_Toc485133025)

[5. Bausteinsicht 8](#_Toc485133026)

[5.1 Ebene 1 8](#_Toc485133027)

[5.1.1 Bausteinname 1 (BlackBox-Beschreibung) 8](#_Toc485133028)

[5.2 Ebene 2 9](#_Toc485133029)

[5.2.1 Bausteinname 1 (Whitebox-Beschreibung) 9](#_Toc485133030)

[6. Laufzeitsicht 10](#_Toc485133031)

[7. Datensicht 11](#_Toc485133032)

[8. Verteilungssicht 12](#_Toc485133033)

[9. Konzepte 13](#_Toc485133034)

[9.1 Frameworks und Entwurfsmuster 13](#_Toc485133035)

[9.1.1 Frameworks 13](#_Toc485133036)

[9.1.2 Entwurfsmuster 13](#_Toc485133037)

[9.2 Persistenz 13](#_Toc485133038)

[9.3 Benutzungsoberfläche 13](#_Toc485133039)

[10. Qualitätsszenarien 14](#_Toc485133040)

[10.1 Bewertungsszenarien 14](#_Toc485133041)

[11. Glossar 15](#_Toc485133042)

# Einführung und Ziele / Fachliche Anforderungen

Als Einführung in das Architekturdokument gehören hierher die treibenden Kräfte, die Software-Architekten bei Ihren Entscheidungen berücksichtigen müssen:

Einerseits die Erfüllung bestimmter fachlicher Aufgabenstellungen der Stakeholder, darüber hinaus aber die Erfüllung oder Einhaltung der vorgegebenen Randbedingungen unter Berücksichtigung der Architekturziele.

## Aufgabenstellung

(engl.: Requirements Overview)

Kurzbeschreibung der fachlichen Aufgabenstellung, Extrakt (oder Abstract) der Anforderungsdokumente.

Verweis auf ausführliche Anforderungsdokumente (mit Versionsbezeichnungen und Ablageorten).

## Qualitätsziele

Inhalt:

Die Hitparade (Top-3 bis Top-5) der Qualitätsziele für die Architektur und/oder Randbedingungen, deren Erfüllung oder Einhaltung den maßgeblichen Stakeholdern besonders wichtig sind.

Gemeint sind hier wirklich Qualitätsziele, die nicht unbedingt mit den Zielen des Projekts übereinstimmen. Beachten Sie den Unterschied.

Als Qualitätsziele findet man in der Praxis oft:

* Verfügbarkeit (availability)
* Änderbarkeit (modifiability)
* Performanz (performance)
* Sicherheit (security)
* Testbarkeit (testability)
* Bedienbarkeit (usability)

Motivation:

Wenn Sie als Architekt nicht wissen, woran Ihre Arbeit gemessen wird, ....

Form:

Einfache tabellarische Darstellung, geordnet nach Prioritäten

## Stakeholder

Inhalt

Eine Liste oder Tabelle der wichtigsten Personen oder Organisationen, die von der Architektur betroffen sind oder zur Gestaltung beitragen können.

Beispiele

Die folgende Tabelle führt Stakeholder auf, die in Projekten relevant sein könn(t)en.

| Stakeholder | Beschreibung |
| --- | --- |
| Management | Linien-Manager, die an dem Projekt beteiligt sind oder es beeinflussen |
| Lenkungskreis | Oberstes Lenkungsgremium des Projektes, ultimative Instanz für Projektentscheidungen |
| Projektleiter | Dem Projektleiter (PL) obliegt die operative Projektdurchführung, d.h. er ist für die Planung, Steuerung und Überwachung des Projektes verantwortlich.  Der PL setzt den Projektauftrag in ausführbare Handlungs-anweisungen um. Er plant das Gesamtprojekt hinsichtlich Kosten, Terminen, Leistungen, Projektorganisation und er erstellt das Projekthandbuch. Bei Störungen des Projektablaufes muss er korrigierende Maßnahmen innerhalb seines Handlungsrahmens und innerhalb seines Auftrages ergreifen bzw. vorschlagen. Der PL führt das Projektteam und berichtet in regelmäßigen Abständen, auf Anforderung oder aus gegebenem Anlass, dem PT. |
| Projektauftraggeber | Der Projektauftraggeber (PAG) ist die Person oder Funktion, die innerhalb des SWM-Konzerns bevollmächtigt ist, das Budget für das Projekt bereitzustellen. Der PAG erteilt und nimmt den Projektauftrag ab und stellt die Projekt-Finanzierung sicher. In Abhängigkeit von der Komplexität und der Bedeutung des Projektes kann der PAG oder die Geschäftsführung im Rahmen des Projektstarts einen übergreifenden Lenkungskreis (LK), in dem Entscheidungsträger der am Projekt beteiligten Geschäftsbereiche vertreten sind, benennen. Der PAG kann seine Projektaufgaben während der Projektlaufzeit an den Projektträger delegieren. |
| Projektträger | Dem Projektträger (PT) obliegt die strategische Projektsteuerung.  Die Rolle kann optional vom PAG in Personalunion wahrgenommen werden. Bei bereichsübergreifenden Projekten wird der PT durch die betroffenen Geschäftsführer benannt.  Er übernimmt in den meisten Fällen auch die Eigentümerfunktion im Projekt und überträgt die fachlichen und konzeptionellen Anforderungen des Unternehmens in Programme und Projekte. Innerhalb der SWM Gesellschaften wird zur Wahrnehmung dieser Aufgaben für Projekte ein Projektträger benannt und der Handlungsrahmen festgelegt. Dieser trifft fachliche Entscheidungen, die über den Entscheidungsrahmen des Projektleiters hinausgehen.  Der PT ist das strategische Entscheidungsgremium im Projekt,  soweit vom PAG beauftragt.  Die Qualitätssicherungs- und Umweltschutzmaßnahmen bezüglich der verschiedenen technischen Prozesse werden vom PT definiert und beauftragt; vom Projektleiter umgesetzt und berichtet. |
| Produktmanager | Verantwortlich für das gesamte Produkt, das aus  Hardware & Software sowie sonstigen Leistungen bestehen kann. |
| Fachbereich | In der Regel die Personengruppe, die die fachlichen Anforderungen formuliert |
| Unternehmens- oder Enterprisearchitekt | u.a. zuständig für strategische Ausrichtung des Anwendungsportfolios und projektübergreifende Richtlinien und Standards |
| Architektur-Abteilung | Gruppe, die Unternehmens-Frameworks und Entwicklungsstandards pflegt |
| Methoden und Verfahren | Verantworten Entwicklungsprozesse und häufig auch die eingesetzte Tool  Hinweis: I.d.R. hat man nicht gleichzeitig Unternehmensarchitekten, eine Architektur-Abteilung und Methoden und Verfahren, sondern  max. 2 davon |
| IT-Strategie | Verantwortlich für die strategische Ausrichtung der IT. Siehe Enterprise-Architekt. |
| QA | Zentrale Test-Abteilung. Verantwortlich für die Qualitätssicherung |
| Systemarchitekt | Verantwortlich für die (technische) Architektur innerhalb  eines Projekts |
| Designer | Zuständig für das Anwendungs-Design. Häufig keine eigene Rolle mehr |
| Entwickler | Software-Entwickler im Projekt. Übernimmt häufig auch Design- und Testaufgaben |
| Tester | Tester im Projekt. Kann aus QA sein, häufig aber unabhängig davon. |
| Konfigurations-& Build-Manager | Zuständig für die Pflege von Repository, Konfigurations-Management und Build. Wird in kleineren Projekten häufig vom Entwickler übernommen. |
| Release-Manager | Verantwortlich für die Erstellung und Auslieferung von Release-Ständen. Koordiniert Releases häufig Projekt- und System-übergreifend |
| Wartungs-Team | Zuständig für die Pflege und Wartung des Systems nach Auflösung des Projekt-Teams |
| Externe Dienstleister | Zusätzliche externe Firmen, die Teile der Anwendung entwickeln. |
| Hardware-Designer | Zuständig für das Hardware-Design (im Embedded-Bereich) |
| Rollout-Manager | Zuständig für die Inbetriebnahme eines Systems oder eines Releases. Rolle wird manchmal vom Release-Manager übernommen |
| Infrastruktur-Planung | Zuständig für Planung und Beschaffung der Infrastruktur (Server, Netzwerk, Router, Switches, Arbeitsplatzrechner, OS, …) |
| Sicherheitsbeauftragter | Verantwortlich für die IT-Sicherheit im Unternehmen |
| Anwender | Nutzer der Anwendung |
| Fach-Administrator | Zuständig für die fachliche Administration der Anwendung. Hat häufig keinen Zugang zu technischen Administrations-Zugängen |
| System-Administrator | Administriert die Anwendung auf technischer Ebene. Hat Zugang zu technischen Administrations-Zugängen |
| Operator | Überwacht den Anwendungsbetrieb, führt Routine-Pflegejobs durch (z.B. Datensicherung, Aufräumen von temporären Verzeichnissen), behebt einfache Fehler im Anwendungsbetrieb |
| Hotline | Häufig auch unter 1st oder 2nd Level Support bekannt. Nehmen Fehlermeldungen auf, helfen in Standardsituationen |
| Betriebsrat | Vertritt die Interessen der Arbeitnehmer |
| Standard-Software-Lieferant | Lieferant von im System eingesetzter Standard-Software.  Unterstützen häufig auch bei Integration und Customizing |
| Verbundene Projekte | z.B. Nachbarprojekte mit gemeinsamen Schnittstellen, übergreifende Schnittstellenprojekte (z.B. EAI/ESB-Projekte) |
| Aufsichtsbehörden, Gesetzgeber, Normierungsgremien | Sind meistens nicht direkt mit dem Projekt verbunden, beeinflussen jedoch durch Ihre Vorgaben die Arbeit bzw. die Lösungsansätze. |
| Weitere externe Stakeholder | z.B. Verbände, Vereine, Mitbewerber, konkurrierende Geschäftsbereiche, Presse. Sind häufig nicht direkt vom Projekt betroffen, beeinflussen Entscheidungen aber dennoch |

|  |  |  |
| --- | --- | --- |
| **Name** | **Rolle** |  |
|  |  |  |
|  |  |  |

# Randbedingungen

Inhalt

Fesseln, die Software-Architekten in ihren Freiheiten bezüglich des Entwurfs oder des Entwicklungsprozesses einschränken.

## Technische Randbedingungen

***Inhalt***

Tragen Sie hier alle technischen Randbedingungen ein. Zu dieser Kategorie gehören Hard- und Software-Infrastruktur, eingesetzte Technologien (Betriebssysteme, Middleware, Datenbanken, Programmiersprachen, ...).

Dieses Kapitel beschreibt spezifische Vorgaben, die die allgemeinen Vorgaben aus Entwicklungshandbüchern etc . konkretisieren.

|  |  |
| --- | --- |
| Hardware-Vorgaben | |
|  |  |
| Software-Vorgaben | |
|  |  |
| Programmiervorgaben | |
|  |  |

***Beispiele***

|  |  |
| --- | --- |
| Randbedingung | Erläuterung |
| Hardware-Infrastruktur | Prozessoren, Speicher, Netzwerke, Firewalls und andere relevante Elemente der Hardware- Infrastruktur |
| Software-Infrastruktur | Betriebssysteme, Datenbanksysteme, Middleware, Kommunikationssysteme, Transaktionsmonitor, Webserver, Verzeichnisdienste |
| Systembetrieb | Batch- oder Onlinebetrieb des Systems oder notwendiger externer Systeme? |
| Verfügbarkeit der Laufzeitumgebung | Rechenzentrum mit 7x24h Betriebszeit?  Gibt es Wartungs- oder Backupzeiten mit eingeschränkter Verfügbarkeit des Systems oder wichtiger Systemteile? |
| Grafische Oberfläche | Existieren Vorgaben hinsichtlich grafischer Oberfläche (Style Guide)? |
| Bibliotheken, Frameworks und Komponenten | Sollen bestimmte „Software-Fertigteile“ eingesetzt werden? |
| Programmiersprachen | Objektorientierte, strukturierte, deklarative oder Regelsprachen, kompilierte oder interpretierte Sprachen? |
| Referenzarchitekturen | Gibt es in der Organisation vergleichbare oder übertragbare Referenzprojekte? |
| Analyse- und Entwurfsmethoden | Objektorientierte oder strukturierte Methoden? |
| Datenstrukturen | Vorgaben für bestimmte Datenstrukturen, Schnittstellen zu bestehenden Datenbanken oder Dateien |
| Programmierschnittstellen | Schnittstellen zu bestehenden Programmen |
| Programmiervorgaben | Programmierkonventionen, fester Programmaufbau |
| Technische Kommunikation | Synchron oder asynchron, Protokolle |
| Betriebssystem und Middleware | Vorgegebene Betriebssysteme oder Middleware |

# Kontext

Inhalt

Die Kontextsicht grenzt das System, für das Sie die Architektur entwickeln, von allen Nachbarsystemen ab. Sie legt damit die wesentlichen externen Schnittstellen fest.

Stellen Sie sicher, dass die Schnittstellen mit allen relevanten Aspekten (was wird übertragen, in welchem Format wird übertragen, welches Medium wird verwendet, ...) spezifiziert wird, auch wenn einige populäre Diagramme (wie z.B. das UML Use-Case Diagramm) nur ausgewählte Aspekte der Schnittstelle darstellen.

Motivation

Die Schnittstellen zu Nachbarsystemen gehören zu den kritischsten Aspekten eines Projektes. Stellen Sie rechtzeitig sicher, dass Sie diese komplett verstanden haben.

Form

* Diverse Kontextdiagramme (siehe folgende Abschnitte)
* Listen von Nachbarsystemen mit deren Schnittstellen.

## Externe Schnittstellen

### Externe Schnittstelle 1

#### Identifikation der Schnittstelle

|  |  |
| --- | --- |
| Name / Bezeichnung der Schnittstelle | <Name der Schnittstelle> |
| Version |  |
| Typ der Schnittstelle |  |
| Anforderungen an die Schnittstelle |  |
| Ablauf der Schnittstelle |  |

# Lösungsstrategie

***Inhalt***

Kurzer Überblick über Ihre grundlegenden Entscheidungen und Lösungsansätze, die jeder, der mit der Architektur zu tun hat, verstanden haben sollte.

# Bausteinsicht

Inhalt

Statische Zerlegung des Systems in Bausteine (Module, Komponenten, Subsysteme, Teilsysteme, Klassen, Interfaces, Pakete, Bibliotheken, Frameworks, Schichten, Partitionen, Tiers, Funktionen, Makros, Operationen, Datenstrukturen...) sowie deren Beziehungen.

Form

Die Bausteinsicht ist eine hierarchische Sammlung von BlackBox- und White-Box-Beschreibungen

## Ebene 1

Das Überblicksbild zeigt das Innenleben des Gesamtsystems in Form der Bausteine 1 - n, sowie deren Zusammenhänge und Abhängigkeiten.

Sinnvoll sind hier auch Beschreibungen der wichtige Begründungen, die zu dieser Struktur führen, insbesondere die Beschreibung der Abhängigkeiten (Beziehungen) zwischen den Bausteinen dieser Ebene.

### Bausteinname 1 (BlackBox-Beschreibung)

* **Zweck / Verantwortlichkeit:**
* **Schnittstelle(n):**
* **Erfüllte Anforderungen**

## Ebene 2

### Bausteinname 1 (Whitebox-Beschreibung)

#### Bausteinname 1.1 (BlackBox Beschreibung)

1. Zweck / Verantwortlichkeit:
2. Schnittstelle(n):
3. Erfüllte Anforderungen:

# 

# Laufzeitsicht

Inhalt

Diese Sicht beschreibt, wie sich die Bausteine des Systems als Laufzeitelemente (Prozesse, Tasks, Activities, Threads, ...) verhalten und wie sie zusammenarbeiten.

Anmerkung: Kriterium für die Auswahl der möglichen Szenarien (d.h. Abläufe) des Systems ist deren *Architekturrelevanz.* Es geht nicht darum, möglichst viele Abläufe darzustellen, sondern eine angemessene Auswahl zu dokumentieren.

Kandidaten sind:

1. Die wichtigsten 3-5 Anwendungsfälle
2. Systemstart
3. Das Verhalten an den wichtigsten externen Schnittstellen
4. Das Verhalten in den wichtigsten Fehlerfällen

Form

Dokumentieren Sie die ausgesuchten Laufzeitszenarien mit UML-Sequenz-, Aktivitäts-, oder Kommunikationsdiagrammen.

# Datensicht

Welche Kategorien von Daten müssen gespeichert werden?

Wo werden die einzelnen Daten gespeichert?

# 

# Verteilungssicht

Inhalt

Diese Sicht beschreibt, in welcher Umgebung das System abläuft. Sie beschreiben die geographische Verteilung Ihres Systems oder die Struktur der Hardwarekomponenten, auf denen die Software abläuft. Sie dokumentiert Rechner, Prozessoren, Netztopologien und Kanäle, sowie sonstige Bestandteile der physischen Systemumgebung. Die Verteilungssicht zeigt das System aus Betreibersicht.

Zeigen Sie in dieser Sicht auch, wie die Bausteine des Systems zu *Verteilungsartefakten* zusammengefasst oder –gebaut werden (engl. deployment artifacts oder deployment units).

Form

Die UML stellt mit Verteilungsdiagrammen (Deployment diagrams) eine Diagrammart zur Verfügung, um diese Sicht auszudrücken. Nutzen Sie diese, evtl. auch geschachtelt, wenn Ihre Verteilungsstruktur es verlangt. (Das oberste Deployment- Diagramm sollte bereits in Ihrer Kontextsicht enthalten sein mit Ihrer Infrastruktur als EINE Black-Box. Jetzt zoomen Sie in diese Infrastruktur mit weiteren Deployment- Diagrammen hinein.

Andere Diagramme Ihrer Hardware-Kollegen, die Knoten und Kanäle darstellen sind hier ebenfalls einsetzbar. Abstrahieren Sie aber auf die Aspekte, die für die Software-Verteilung relevant sind.

# Konzepte

Inhalt:

Die folgenden Kapitel sind Beispiele für übergreifende Aspekte.

Falls einige der Aspekte für Ihr Projekt nicht wichtig sind oder nicht zutreffen, so halten Sie *diese* Information ebenfalls fest, anstatt das Kapitel zu löschen.

## Frameworks und Entwurfsmuster

### Frameworks

Welche Frameworks werden im Anwendungssystem verwendet?

Welche Funktionen decken sie jeweils ab?

### Entwurfsmuster

Welche Entwurfsmuster werden verwendet?

## Persistenz

Persistenz (Dauerhaftigkeit, Beständigkeit) bedeutet, Daten aus dem (flüchtigen) Hauptspeicher auf ein beständiges Medium (und wieder zurück) zu bringen.

## Benutzungsoberfläche

IT-Systeme, die von (menschlichen) Benutzern interaktiv genutzt werden, benötigen eine Benutzungsoberfläche. Das können sowohl grafische als auch textuelle Oberflächen sein.

Welcher Art ist die Benutzeroberfläche? Fat Windows Client? Web-Client?

Kann der Anwender die Oberfläche konfigurieren?

# Qualitätsszenarien

Dieses Kapitel fasst alles zusammen, was Sie zur systematischen Bewertung Ihrer Architektur gegen vorgegebene Qualitätsziele benötigen.

## Bewertungsszenarien

Inhalt

Szenarien beschreiben, was beim Eintreffen eines Stimulus auf ein System in bestimmten Situationen geschieht. Sie charakterisieren damit das Zusammenspiel von Stakeholdern mit dem System. Szenarien operationalisieren Qualitätsmerkmale und machen sie messbar.

Wesentlich für die meisten Software-Architekten sind zwei Arten von Szenarien:

1. Nutzungsszenarien (auch genannt *Anwendungs- oder Anwendungsfallszenarien*) beschreiben, wie das System zur Laufzeit auf einen bestimmten Auslöser reagieren soll. Hierunter fallen auch Szenarien zur Beschreibung von Effizienz oder Performance. Beispiel: Das System beantwortet eine Benutzeranfrage innerhalb einer Sekunde.
2. Änderungsszenarien beschreiben eine Modifikation des Systems oder seiner unmittelbarer Umgebung. Beispiel: Eine zusätzliche Funktionalität wird implementiert oder die Anforderung an ein Qualitätsmerkmal ändert sich.



**Abbildung: Schematische Darstellung von Szenarien (nach [Bass+03])**

Szenarien bestehen aus folgenden wesentlichen Teilen (hier zitiert aus [Starke05], die ursprüngliche Gliederung stammt aus [Bass+03]):

1. Auslöser (*stimulus*): beschreibt eine spezifische Zusammenarbeit des (auslösenden) Stakeholders mit dem System. Beispiele: Ein Benutzer ruft eine Funktion auf, ein Entwickler programmiert eine Erweiterung, ein Administrator installiert oder konfiguriert das System.
2. Quelle des Auslösers (*source*): beschreibt, woher der Auslöser kommt. Beispiele: intern oder extern, Benutzer, Betreiber, Angreifer, Manager.
3. Umgebung (*environment*): beschreibt den Zustand des Systems zum Zeitpunkt des Auslösers. Befindet sich das System unter Normal- oder Höchstlast? Ist die Datenbank verfügbar oder nicht? Sind Benutzer online oder nicht? Hier sollten Sie alle Bedingungen beschreiben, die für das Verständnis des Szenarios wichtig sind.
4. Systembestandteil (*artifact*): beschreibt, welcher Bestandteil des Systems vom Auslöser betroffen ist. Beispiele: Gesamtsystem, Datenbank, Webserver.
5. Antwort (*response*): beschreibt wie das System durch seine Architektur auf den Auslöser reagiert. Wird die vom Benutzer aufgerufene Funktion ausgeführt? Wie lange benötigt der Entwickler zur Programmierung? Welche Systemteile sind von Installation/Konfiguration betroffen?
6. Antwortmetrik (*response measure*): beschreibt, wie die Antwort gemessen oder bewertet werden kann. Beispiele: Ausfallzeit in Stunden, Korrektheit Ja/Nein, Änderungszeit in Personentagen, Reaktionszeit in Sekunden.

Form

Entweder tabellarisch oder als Freitext. Sie sollten die Bestandteile (Quelle, Umgebung, Systembestandteil, Antwort, Antwortmetrik) explizt kenntlich machen.

# 

# Glossar

Inhalt

Die wichtigsten Begriffe der Software-Architektur in alphabetischer Reihenfolge

|  |  |
| --- | --- |
| Begriff | Definition |
|  |  |
|  |  |