**Technische Dokumentation**

für

**Eiskonzentration Beobachtung**

Version 1.0

verfasst von

|  |  |  |
| --- | --- | --- |
| Ehab Fadl |  | [fadl@th-brandenburg.de](mailto:fadl@th-brandenburg.de) |
| Oussama Rabhi |  | <Email> |
| Arjit Kumar Singh |  | [singha@th-brandenburg.de](mailto:singha@th-brandenburg.de) |
| <Name> |  | <Email> |
| <Name> |  | <Email> |

|  |  |
| --- | --- |
| Dozent: | Prof. Dr. rer. nat. Gabriele Schmidt |
| Lehrveranstaltung: | Softwarearchitektur und Qualitätssicherung |
|  |  |
| Datum: | 10/12/2019 |
|  |  |

Inhaltsverzeichnis

Inhaltsverzeichnis iii

Historie der Dokumentversionen iv

1. Komponenten (Statische Sicht) 4
   1. Komponenten 4
      1. Komponente <Name> 4
   2. Schnittstellen 4
      1. Schnittstelle<Name> 4
   3. Regeln 5
2. Quellen VII

Historie der Dokumentversionen

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Datum** | **Autor (en)** | **Änderungsgrund/Bemerkungen** |
| *0.1* | *00/00/00* | *<Hauptautor genügt>* | *<Informationen über die Revision (z.B. Ersterstellung, Aktualisierung, bereit zur Abnahme, abgenommen, …)>* |
|  |  |  |  |
|  |  |  |  |

<In dieser Vorlage finden Sie Texte, die von den "<>" Symbolen beschränkt sind. Dieser Text (in Kursivschrift) soll Sie durch die Vorlage führen und Erklärungen zu den verschiedenen Abschnitten in diesem Dokument liefern.

Bitte stellen Sie sicher, dass Sie alle Kommentare vor dem Absenden/Einreichen des Dokuments löschen.>

**Einführung**

Im Projekt ,,Eiskonzentration Beobachtung” geht es um die Beobachtung des Eisinhalts in der Atmosphäre.

Die Stationen werden von den Mitarbeitern besucht und die aktuelle Werte notiert, um die Eiskonzentration in der Atmosphäre zu messen. Diese Werte werden dann in die Software eingegeben.

Diese Angaben werden von einem Mitarbeiter beobachtet und analysiert. Jede Station hat einen Zielwert. Wenn eine Abweichung von weniger als 10% zwischen dem Ziel- und aktuellen Wert vorliegt, wird die Abweichung im Rot dargestellt. Die Abweichung wird im Grün dargestellt, falls die Abweichung über 5% liegt. Dieser ausgewählte Standort wird analysiert, um einen Erfolgsfaktor zu finden. Die Abweichung wird im Schwarz dargestellt, wenn der Zielwert und aktuelle Wert im Bereich zwischen -10% und 5% liegen.

Die Möglichkeit wird vom Programm einem Mitarbeiter den aktuellen Wert und das Datum einzutragen gegeben. Die Benachrichtigungen bei Einfügung neuer Stationen ins System und Aktualisierung des Eisinhaltes werden vom System dem Nutze gegeben.

Das Ziel dieses Projekt ist ein geeignetes Architekturmodel zu finden, um unsere Applikation mit der Berücksichtigung unserer Qualitätsanforderungen zu entwickeln.

**Qualitätsanforderungen**

**Usability Varianz I**

Priorität: Sehr wichtig

Nach dem Eintragen eines Neuen Wert, wird die Varianz auf dem GUI dem Mitarbeiter erhalten. Der in View eingetragene Wert wird durch den Presenter zu dem Model geschickt. Die Varianz wird während der Laufzeit im Model berechnet, in der Datenbank gleichzeitig gespeichert und in wenigen Sekunden mit der richtigen Farbe angezeigt.

**Usability Varianz II**

Priorität: Sehr wichtig

Die Art der Abweichung wird vom Mitarbeiter zur Laufzeit bestimmt und beurteilt. Die Abweichung wird in drei Farben angezeigt und zwar Rot, wenn die Abweichung mehr als 10% niedriger als Zielwert (negativ) ist. Grün, wenn die Abweichung mehr als 5% hoher als Zielwert (positiv) ist und Schwarz anderen Falls.

**Usability Station**

Priorität: Sehr wichtig

Nach einer Meldung eines neuen Station während der Laufzeit des Systems wird eine neue Station bald durch das System generiert. Nach Erstellung einer neuen Station durch das System wird eine Benachrichtigung an die View geschickt. Der Nutzer kann diese Station jetzt anzeigen und verwenden.

**Modifiability GUI I**

Priorität: Sehr wichtig

Die Trennung der Logik und View hilft bei der Erstellung einer neuen GUI durch den Entwickler bzw. Wartungsentwickler während der Implementierungsphase/Wartungsphase. Die neue GUI wird auch getestet und dokumentiert. Das ist innerhalb eines Tages zu erfolgen möglich, ohne die Logik zu ändern.

**Modifiability GUI II**

Priorität: Sehr wichtig

Wie oben geschrieben, es ist auch möglich eine neue GUI auf ein neues Gerät zu erstellen. Das hat keine Wirkung auf Logik und wird innerhalb eines Tages erledigt.

**Modifiability Logik**

Priorität: Sehr wichtig

Der Entwickler/Wartungsentwickler kann eine neue Funktionalität währen der Implementierungsphase/Wartungsphase hinzufügen. Die neue Funktionalitäten werden nur im Model hinzufügt. Die andere Komponenten werden nicht verändert

**Availability GUI**

Priorität: Wichtig

Wenn eine GUI beim normalen Betrieb ausgefallen ist, funktionieren andere GUIs einfach weiter. Dieser Ausfall wird keine Auswirkung auf das Model haben. Das GUI wird in einem Werktag wieder funktionieren.

**Availability System**

Priorität: Wenig wichtig

Wenn das ganze System beim normalen Betrieb ausfällt, wird Reparatur im Model implementiert. Das System funktioniert weiter innerhalb eines halben Tages.

**Performance Station**

Priorität: Wichtig

Wenn eine neue Station während normalem Betrieb erstellt wird, werden alle Nutzer gleichzeitig durch eine Benachrichtigung auf ihren GUIs informiert. Das erfolgt in weniger Sekunden.

**Performance Varianz**

Priorität: Sehr wichtig

Die Varianz wird in allen GUIs wie bei der Station gleichzeitig angezeigt.

**Testability GUI**

Priorität: Sehr wichtig

Ein Unit-Test wird vom Entwickler während Implementierung/Wartungsphase implementiert. Damit lauft ein Test für die GUI automatisch durch. Das erfolgt innerhalb eines Tages.

# Komponenten (Statische Sicht)

<Modul-/Implementierungssicht, Physikalische Sicht >

## Komponenten

<Überblick durch Komponentendiagramm mit Schnittstellen und Bezug zu einer Mehrschichtenarchitektur >

### Komponente <Name>

<Die Komponente und ihre Hauptfunktionen sollten klar werden, dabei werden die wichtigen Designentscheidungen begründet (Diskussion Stärken/Schwächen). Ggf. werden Modifikationen gegenüber dem gewählten Architekturstilen/Architekturmuster angegeben

Für die jeweiligen Komponenten bietet ggf. ein Klassendiagramm Übersicht.

Bitte bei verwendeten (Entwurfs-)Mustern, diese beschreiben und deren Verwendung begründen, d.h. hier mehr ins Detail gehen.>

## Schnittstellen

### Schnittstelle<Name>

**<** Schnittstellen soll sehr detailliert beschrieben werden:

* Funktionsaufruf und Parameterleiste
  + Namen und Zweck der Funktionen/Methoden
  + Name, Bedeutung und Wertebereiche von Übergabe- und Rückgabeparametern
  + Verhalten der Komponente bei Nutzung einer Methode. Dieses Verhalten kann sich nur über eine Schnittstellen der Komponente zeigen.
* Ausnahmebehandlung,
* Übertragungsrichtung  
  ggf. Unterscheidung Kontrollfluß und Datenfluß
* ggf. Synchronisationsmechanismus
* wichtige Designentscheidungen begründen (Diskussion Stärken/Schwächen)

Schnittstellen können durch eine Schnittstellenspezifikation beschrieben werden, z. B. gemäß IDL von Corba:

interface Name

{ attribute Typ Attribut-Name;

...

void Operation()...;

...

};

Signatur einer Operation:

[oneway] <Ergebnistyp> <Operationsname>   
 (in Typ1 Parameter1,   
 out Typ2 Parameter2,   
 inout Typ3 Parameter3, ...)

[raises (Ausnahme1, ..., AusnahmeM)]

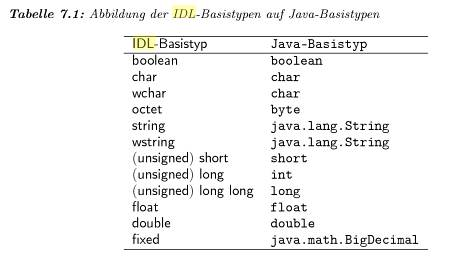
[context (Kontextname1,..., KontextnameK)]

Komplexe Typen als struct definiert werden

struct Typname{

Typ Attribut-Name;

};



(Quelle : Oliver Haase, Kommunikation in verteilten Anwendungen: Einführung in Sockets, Java RMI, CORBA und Jini, Oldenbourg Wissenschaftsverlag, 2008)

>

## Regeln

<Welche Regeln/ generellen Prinzipien gelten für die Kommunikation/Interaktion der Komponenten? Regeln auflisten, begründen, indem Sie auf Qualitätsanforderungen bezogen werden.>

Kommunikation (Dynamische Sicht)< Beschreiben Sie in zwei Sequenzdiagrammen interessante Abläufe der Funktionalität des Systems>

Technologie

< Beschreiben Sie die gewählten Technologien, so dass klar ist, wie die Komponenten und deren Interaktion umgesetzt wurde, z. B. Programmiersprachen, Kommunikationsprotokolle, Synchronisation, Verteilung auf Hardware etc. Ggf Verteilungsdiagramm

Bewerten Sie, wie sich die Technologie für die Architektur eignet>

# Quellen