**Technische Dokumentation**

für

**<Projekt>**

Version <X.X>

verfasst von

|  |  |  |
| --- | --- | --- |
| <Name> |  | <Email> |
| <Name> |  | <Email> |
| Kamgo Chetchom Romeo Landry |  | kamgoche@th-brandenburg.de |
| <Name> |  | <Email> |
| <Name> |  | <Email> |

|  |  |
| --- | --- |
| Dozent: |  |
| Lehrveranstaltung: |  |
|  |  |
| Datum: |  |
|  |  |

Inhaltsverzeichnis

[Inhaltsverzeichnis iii](#_Toc434394800)

[Historie der Dokumentversionen iv](#_Toc434394801)

[1 Einführung 1](#_Toc434394802)

[2 Funktionalität des Systems 3](#_Toc434394803)

[3 Qualitätsanforderungen 5](#_Toc434394804)

[4 Architektur 7](#_Toc434394805)

[5 Komponenten (Statische Sicht) 9](#_Toc434394806)

[5.1 Komponenten 9](#_Toc434394807)

[5.1.1 Komponente <Name> 9](#_Toc434394808)

[5.2 Schnittstellen 9](#_Toc434394809)

[5.2.1 Schnittstelle<Name> 9](#_Toc434394810)

[5.3 Regeln 10](#_Toc434394811)

[6 Kommunikation (Dynamische Sicht) 11](#_Toc434394812)

[7 Technologie 14](#_Toc434394813)

[Quellen XV](#_Toc434394814)

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

<Was ist die Zielaufgabe des Projektes? Konzeptuelle Sicht - sehr knapp nur zum Verständnis >

# Das Ziel unseres Projekts ist die Einrichtung einer Anwendung anhand drei Einheiten nämlich eines Datenmodels, einer Präsentation und einer Programmsteuerung.

# Es geht darum, das Überwachen der Menge an Eis Partikel in der Atmosphäre zu realisieren. Dafür muss das System neue Stationen auf zufällige Weise hinzufügen.

# Zwei User-interfaces (GUI) werden von Staffeln verwaltet. Einer davon dient zur Verwaltung von der Station bzw. eine Station auszuwählen und dann deren aktuellen Wert einzugeben Schließlich ist die Berechnung der Varianz vom System ausgeführt und zurückgegeben. Das zweite dient zur Beobachtung von existierenden Stationen.

# Funktionalität des Systems

<Beschreibung der Funktionalität des Systems im Überblick, z. B. mit Hilfe eines Use-Case-Diagramms>

## II.1. Use-cases

### Add Station

|  |  |
| --- | --- |
| Name of the use case | Add Station |
| Short description | When a new station was added to the system, this is indicated on the GUI. |
| Primary actors | System (Logic) |
| Secondary actors | - |
| Stakeholder | - |
| Trigger | A new station is available |
| Precondition | The System is running in normal mode. |
| Result | The new station will be displayed and values may be entered. |
| Postcondition | New station is integrated like any other station in the overall system |
| Basic flow | 1. The system creates randomly a new station. 2. The system is notifying all display components of the   new station   1. The user can select new station and 2. can use it. |
| Extensions: | - |

### Varianzberechnung

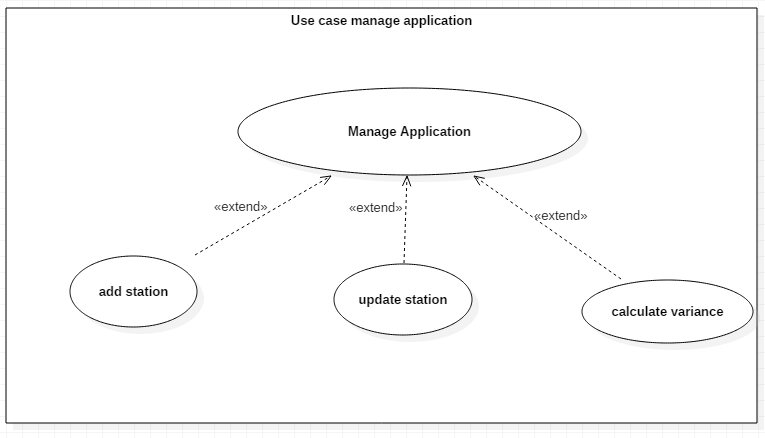
|  |  |
| --- | --- |
| Name of the use case | Caculate variance |
| Short description | Calculate the variance betweenTraget and actual |
| Primaryactors | System (Logic) |
| Secondaryactors | - |
| Stakeholder | - |
| Trigger | Station selected |
| Precondition | Added station |
| Result | Variance calculated |
| Postcondition | Variance iscalculated and Resultisshowed |
| Basic flow | 1. Read Target 2. Enter actual 3. Calculate the variance |
| Extensions: | - |

### Update Station

|  |  |
| --- | --- |
| Name of the use case | Update Station |
| Short description | Modify the datas. |
| Primaryactors | System (Logic) |
| Secondaryactors | - Stuffers |
| Stakeholder | - |
| Trigger | A new station isavailable |
| Precondition | The System is running in normal mode. |
| Result | The Datas (idStation/Date/actual) are modified |
| Postcondition | Modified station |
| Basic flow | 1. Modify the datas 2. Save the new datas |

## II.2.Diagram Use Cases

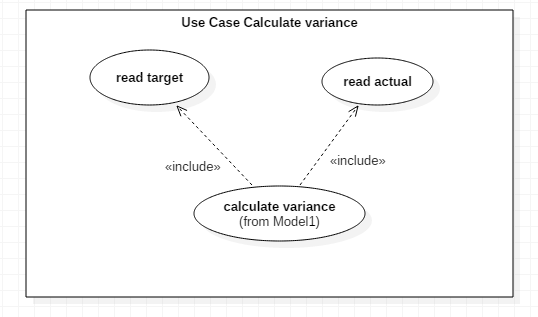
### II.2.1.Use case General



1.Use Case General

Mit `Manage Applikation 'können wir eine neue Station hinzufügen, um eine Station zu aktualisieren, die wir hinzugefügt haben, um die Variantenhexe zu berechnen, ist der Unterschied zwischen` Target' und 'Aktual' und um eine Benachrichtigung nach jeder Änderung anzuzeigen.

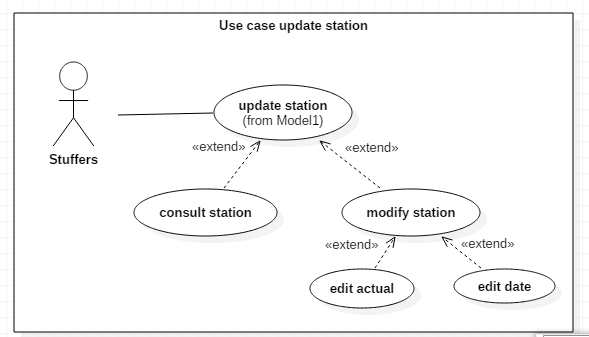
### II.2.2.Use Case Calculate variance



2.Use Case Calculate variance

"Calculate Varience" muss `Target´ und `Aktual´ erhalten, um die Differenz zwischen ihnen zu berechnen, um `Varience´ zu erhalten.

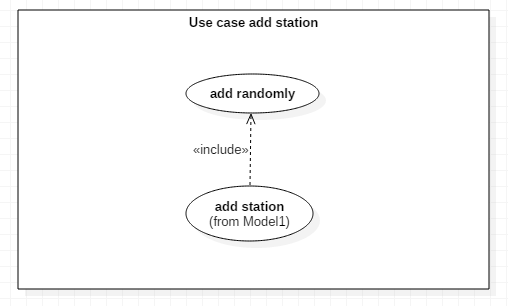
### II.2.3.Use case update station



3.Use Case update Station

Mit Update Station können Stuffers den aktuellen Station abfragen oder ändern..

### II.2.4.Use Case add station



4.Use Case add Station

Mit Add Station wird eine Station zufällig zur Liste hinzugefügt

# Qualitätsanforderungen

<Beschreibung der Qualitätsanforderungen an das System in Form von Szenarien. (Referenz auf gegebenes Material möglich) Diese Szenarien sollen durch entsprechende Taktiken in der Architektur umgesetzt werden. >

In diesem Teil werden einige konkrete Qualität Attribute beschreiben, bzw. Benutzbarkeit (die beide Abweichungen), Modifizierbarkeit, Testbarkeitund Performance.

**Benutzbarkeit**

In der Tat soll einen Benutzer, der schon im System hinzugefügt wurde, die Möglichkeit haben, die Abweichung zu lesen. Je nachdem dieser Abweichung groß ist, muss der Benutzer die Type der Abweichung in 0,5 Sekunde ermitteln. Außerdem die Ergebnisse müssen in verschiedene Farbe dargestellt werden.

**Performance und Benutzbarkeit**

Jede neue Station, die das System erstellt, soll in maximal 1 Sekunden nutzbar sein. Hingegen während einer normalen Nutzungen mit 200 Benutzer, die Benutzer müssen in maximal 5 Sekunden informieren sein.

**Modifizierbarkeit**

Während der Implementierung Phase oder der Instandhaltung Phase muss der Entwickler die Möglichkeit haben, eine neue Funktionalität in einem Tage hinzuzufügen, ohne dass eine Auswirkung auf GUI habt.

**Testbarkeit**

Die Anwendung muss einfach besonders das GUI getestet werden. Also während der Implementierung Phase muss der Entwickler in einem tage Test von GUI schreiben und das automatische laufen lassen.

# Architektur

<Struktur/Architektur als Überblick dokumentiert (Logische Sicht, Konzeptuelle Sicht), dabei werden die wichtigen Architekturentscheidungen begründet:

Klassifikation der Architektur nach einem Architekturstil/Architekturmuster. Begründung der Wahl des Stils/Musters (Stärken/Schwächen) Welche Szenarien werden durch die Architektur (Stil/Muster) unterstützt? (ggf. auch Zuordnung der Szenarien zu Taktiken)>

# 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 Architekturstil/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)]

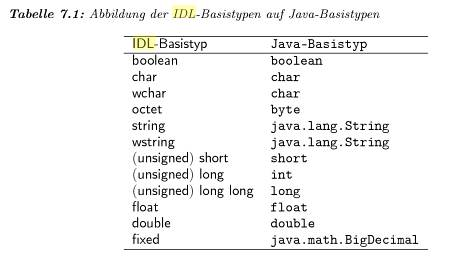
[context (Kontextname1,..., KontextnameK)]

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>

# TestFX

TestFx ist ein „library“ für simplen Test von JavaFx Applikation und Komponenten.

Quellen