# InGrid

## Innovative Grid-Entwicklungen für ingenieurswissenschaftliche Anwendungen

## 3. Bericht AP 2.5: Spezifikation und Softwareentwurf der Benutzerschnittstelle Ergebnisauswertung und Visualisierung


**Projektpartner:**

WASY Gesellschaft für wasserwirtschaftliche Planung und Systemforschung mbH
Waltersdorfer Straße 105
12526 Berlin

**Bearbeiter:**

Dr. Junfeng Luo
Dr. André Pönitz
Dipl.-Ing. Olaf Arndt
Dipl.-Hyd. Udo Junghans
Dipl.-Geol. Kerstin Kernbach

## Inhaltsverzeichnis



## Abbildungsverzeichnis

# 1 Ergebnisauswertung zu den InGrid-Anwendungsszenarien

## 1.1 Beispielmodell Schleuse Storkow

Da Zielgröße bzw. Ergebnis im vorliegenden Beispielmodell die Einhaltung von definierten Wasserständen in Abhängigkeit der Drainagekapazität sein soll, wäre zur Ergebnisauswertung folgendes denkbar:

1. Statistik der simulierten GW-Stände (d. h. Minimal-, Maximal-, Mittelwert sowie Standardabweichung) an ausgewählten GW-Messstellen (GWM), die im FEFLOW-Modell als Observationspunkte gemäß ihrer (horizontalen und vertikalen) Lage festgelegt sind.
2. Häufigkeit der Überschreitung vordefinierter Wasserstandsgrenzwerte in den ausgewählten GW-Messstellen und Registrierung geänderter Parameter einschließlich deren Wertbereiche, welche zur Überschreitung geführt haben.

Ergebnisdaten können in tabellarischer Form zur Auswertung ausgegeben werden, z. B. in folgendem Format

| GWM-Name | Min | Mittel | Max | Std. |
|---|---|---|---|---|
| GWM1 | min1 | mittel1 | max1 | std1 |
| GWM2 | min2 | mittel2 | max2 | std2 |

…

| GWM-Name | Grenzwert* | Überschreitung** | Parameter*** | P-Wert |
|---|---|---|---|---|
| GWM1 | grenz1 | Anzahl1/Gesamt [%] | P-Name | Pwert1 |
| GWM2 | grenz2 | Anzahl2/Gesamt [%] | P-Name | Pwert2 |

…

* Denkbar wäre, dass die Grenzwerte auch als Intervalle vorgeben werden können. Dementsprechend ist die Überschreitungshäufigkeit hinsichtlich vorgegebener Intervalle zu registrieren bzw. auszugeben. Wenn man für eine GWM gleich mehrere Intervalle vorgibt, erhält man die Verteilung der simulierten Wasserstände in der GWM.

** Dies entspricht der Überschreitungshäufigkeit der jeweiligen Grenzwerte bzgl. Gesamtmodellberechnungen im Lauf der Parametermodifikation.

*** Zu registrieren sind neben den P-Werten auch die Elemente, bei denen die Parameter geändert wurden. Generell sind auch Min-, Mittel- und Max-Werte der geänderten Parameter zu registrieren.

Im Falle instationärer Simulationen sind neben den o. g. Werten auch die Berechnungszeitpunkte zu registrieren, bei denen die Max- bzw. Min-Werte aufgetreten sind. Es kann erforderlich sein, dass die Zeitpunkte für alle Überschreitungsereignisse registriert werden müssen. Als weitere Möglichkeit wäre die Häufigkeit von Überschreitungsereignissen bzgl. z. B. Kalendermonaten oder Kalendertagen zu erfassen.

Die Ergebnisse der Statistik bzw. Häufigkeit können z. B. als Histogramm bzgl. der geografischen Lage der GWM räumlich dargestellt werden.

## 1.2 Beispielmodell Drömling

Innerhalb der Projektbearbeitung soll für den in Niedersachsen und Sachsen-Anhalt gelegenen Naturpark Drömling auf Basis der Modelle WBalMo und FEFLOW eine umfassende Wasserbilanz aufgestellt werden. Hierfür müssen alle bilanzwirksamen

Einflussgrößen erfasst und berücksichtigt werden. Dies schließt auch die anthropogenen Einwirkungen auf den Wasserhaushalt ein. Dazu gehören unter anderem Grundwasserentnahmen durch verschiedene Nutzer, wie z. B. Wasserwerke, Beregnungsverbände und Entnahmen für industrielle Zwecke.

Die stationäre Modellkalibrierung ergab eine hohe Sensibilität des Modells für die Durchlässigkeitsbeiwerte. Daher wären eine Sensitivitätsanalyse dieses Parameters und sein Einfluss auf die Ausdehnung der Einzugsgebiete sowie der durch den Stauer fließenden Grundwassermengen denkbar.

Die Anpassung der $k_f$-Werte sollte innerhalb vorher definierter Bereiche sowie global für das gesamte Modellgebiet für die verschiedenen Horizonte (Layer) möglich sein. Die Änderung der $k_f$-Werte sollte innerhalb zu definierender Wertebereiche für einzelne Layer erfolgen.

Zur Ergebnisauswertung sollten folgende Daten bereitgestellt werden:

1. Statistik der simulierten Grundwasserstände an bestimmten Förderbrunnen und Grundwassermessstellen (d. h. Minimal- Maximal-, Mittelwert sowie Standardabweichung und Histogramm mit der Differenz zwischen berechnetem und gemessenem Grundwasserstand, s. Abbildung 1-1), die als Observationspunkte im FEFLOW-Modell definiert sind.

Die Statistikdaten können in tabellarischer Form zur Auswertung ausgegeben werden, z. B. in folgendem Format:

| Polygon | $k_f$-Wert | Min | Mittel | Max | Standardabweichung |
|---|---|---|---|---|---|
| 1 | $10^{-4}$ m/s | -0,48 m | 0,00 m | 0,48 m | 0,13 m |
| 2 | $10^{-5}$ m/s | -0,5 m | 0,15 m | 0,75 m | 0,25 m |

sowie

2. Automatische Bahnlinienberechnung für vordefinierte Brunnen und jeweilige Filterstrecken mit Ausweisung der Umhüllenden (Ausgabe als shp-Dateien, s. Abbildung 1-2),
3. Bilanzierung von durch den Stauer fließenden Grundwassermengen innerhalb der Bahnlinienumhüllenden für eine vordefinierte Slice, Ausgabe als ASCII-Datei bzw. in tabellarischer Form, z. B.:

| Polygon | $k_f$-Wert | EZG Br. X | Fläche | downward Flux | upward Flux |
|---|---|---|---|---|---|
| 1 | $10^{-4}$ m/s | 1 | 50 km² | 500 m³/d | 100 m³/d |
| 1 | $10^{-4}$ m/s | 2 | 30 km² | 250 m³/d | 50 m³/d |
| 2 | $10^{-5}$ m/s | 1 | 15 km² | 600 m³/d | 850 m³/d |

….

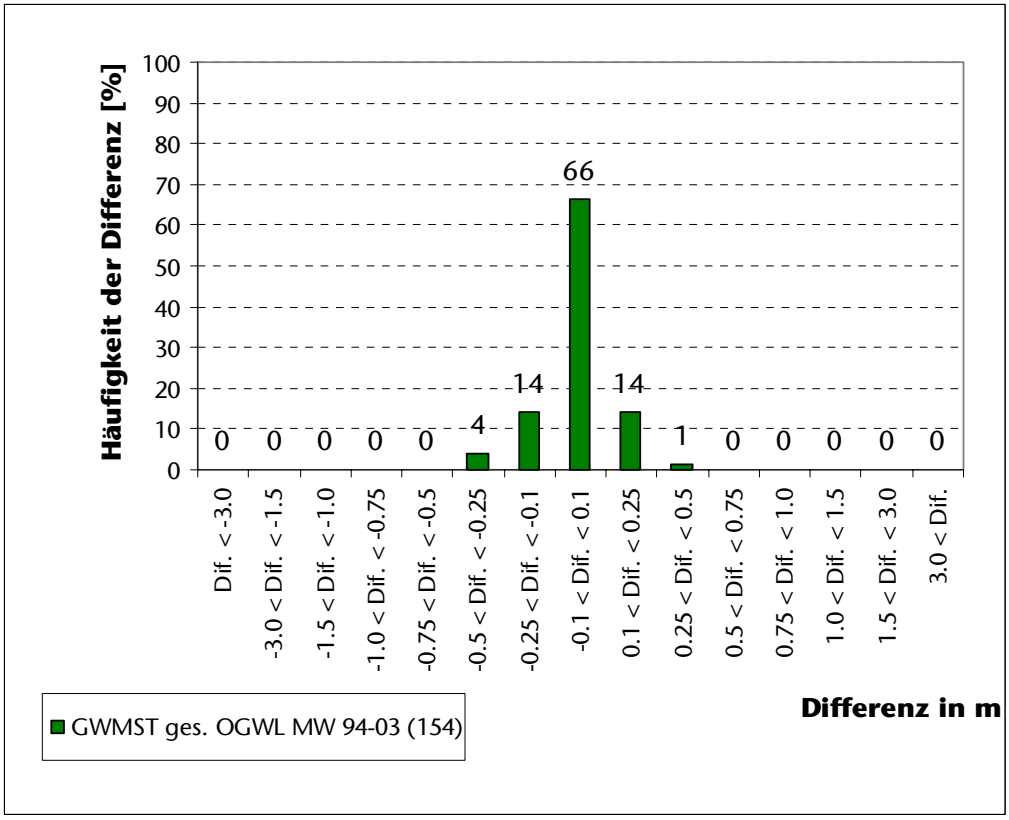


**Abbildung 1-1: Beispiel für zu erstellendes Histogramm, dargestellt ist die Differenz zwischen berechnetem und gemessenem Grundwasserstand an bestimmten Grundwassermessstellen**

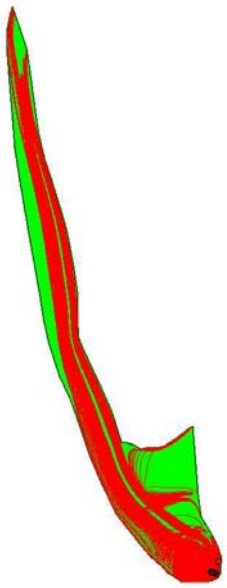

**Abbildung 1-2: Beispiel für berechnete Bahnlinien und Bahnlinienumhüllende**

## 1.3 Beispielmodell GW-Modell WW Tegel

Ziel der Variantenuntersuchung war, eine optimierte Betriebsweise der GW-Entnahme des Wasserwerkes (durch Umverteilung der festgelegten Gesamtfördermenge auf Einzelförderbrunnen) unter Berücksichtigung einer zu bestimmenden GWA-Menge auszuarbeiten. Die Schadstofffahne aus den vorhandenen Altlasten-/Altlastenverdachts-standorten soll durch vorhandene Sicherungsbrunnen wie bisher erfasst werden.

Demnach wäre zur Ergebnisauswertung folgendes denkbar:

1. Statistik der simulierten Konzentration (d. h. Minimal-, Maximal-, Mittelwert sowie Standardabweichung) in bestimmten Förderbrunnen bzw. in ausgewählten GW-Messstellen (GWM), die im FEFLOW-Modell als Observationspunkte gemäß ihrer (horizontalen und vertikalen) Lage festgelegt sind.
2. Häufigkeit der Einhaltung vordefinierter Konzentrationsgrenzwerte in den Förderbrunnen bzw. GW-Messstellen und Registrierung geänderter Parameter einschließlich deren Wertbereiche, welche zur Einhaltung geführt haben.

Ergebnisdaten können in tabellarischer Form zur Auswertung ausgegeben werden, z. B. in folgendem Format

| Name | Min | Mittel | Max | Std. | Parameter*** | P-Wert**** |
|---|---|---|---|---|---|---|
| Br1 | min1 | mittel1 | max1 | std1 | P-Name | Pwert1 |
| Br2 | min2 | mittel1 | max2 | std2 | P-Name | Pwert2 |
| … | | | | | | |
| GWM1 | min1 | mittel1 | max1 | std1 | | |
| GWM2 | min2 | mittel2 | max2 | std2 | | |
| … | | | | | | |

| GWM-Name | Grenzwert* | Einhaltung** | Parameter*** | P-Wert**** |
|---|---|---|---|---|
| Br1 | grenz1 | Anzahl1/Gesamt [%] | P-Name | Pwert1 |
| Br2 | grenz2 | Anzahl2/Gesamt [%] | P-Name | Pwert2 |
| … | | | | |
| GWM1 | grenz1 | Anzahl1/Gesamt [%] | P-Name | Pwert1 |
| GWM2 | grenz2 | Anzahl2/Gesamt [%] | P-Name | Pwert2 |
| … | | | | |

* Denkbar wäre, dass man die Grenzwerte auch als Intervalle vorgeben kann. Dementsprechend ist die Einhaltungshäufigkeit hinsichtlich vorgegebener Intervalle zu registrieren bzw. auszugeben.

** Dies entspricht der Einhaltungshäufigkeit der jeweiligen Grenzwerte bzgl. Gesamtmodellberechnungen im Lauf der Parametermodifikation.

*** Zu registrieren sind neben den P-Werten auch die Elemente, bei denen der Parameter geändert wurde.

**** Der P-Wert kann einer Kombination der Parametermodifikationen (bspw. die Umverteilung der Fördermenge auf verschiedene Brunnen) entsprechen, die zur Einhaltung der Grenzwerte geführt hat. Generell sind auch Min-, Mittel- und Max-Werte der geänderten Parameter zu registrieren.

In Falle instationärer Simulationen sind neben o. g. Werten auch die Berechnungszeitpunkte zu registrieren, bei denen die Max- bzw. Min-Werte aufgetreten sind. Es kann erforderlich sein, dass die Zeitpunkte für alle Einhaltungsereignisse registriert

werden müssen oder die Häufigkeit der Einhaltungsereignisse z. B. bzgl. Kalendermonate bzw. Kalendertage benötigt wird.

Man kann die Ergebnisse der Statistik bzw. Häufigkeit (z. B. als Histogramm) bzgl. der geografischen Lage der Förderbrunnen bzw. GWM räumlich darstellen.

## 1.4 Beispielmodell GW-Modell Flughafen Schönefeld

Sensivitätsuntersuchungen dieses Beispiels dienen zur Quantifizierung der Unsicherheit der durch Modellsimulation ermittelten GW-Fördermenge zur GW-Absenkung für die Bauphase der geplanten Bahntrasse auf dem Flughafengelände.

Die Lage der betrachteten Bahntrasse sowie der entsprechenden Bauabschnitte ist in Abbildung 1-3 als Übersicht gegeben.

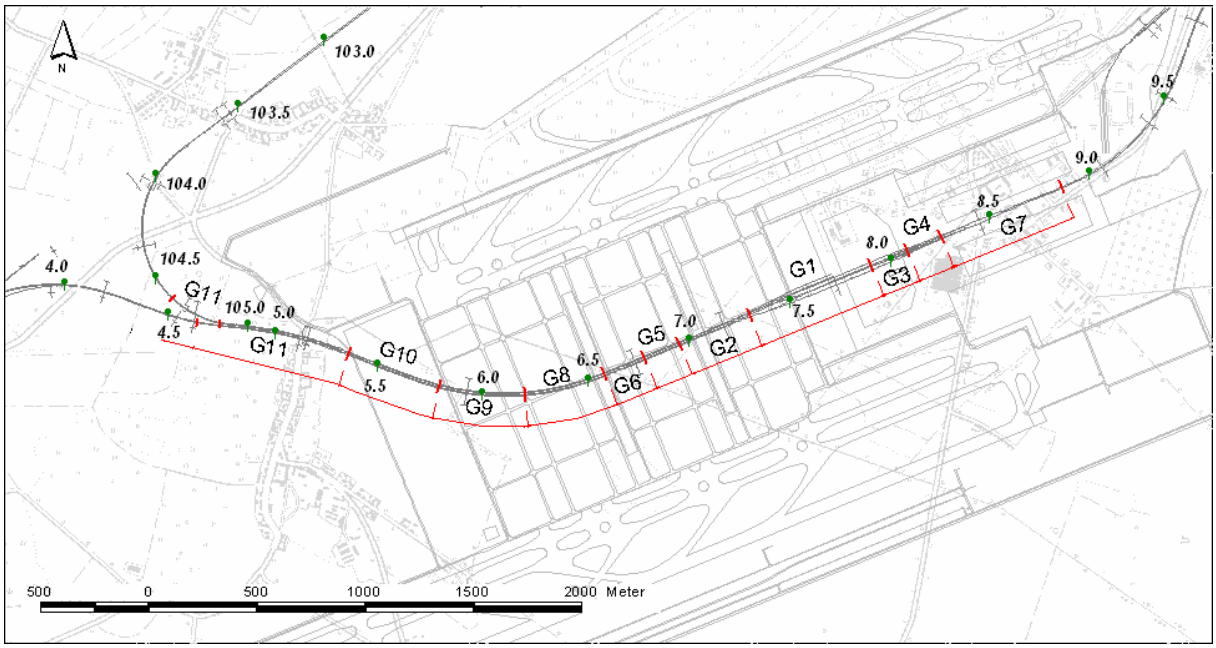


**Abbildung 1-3: Lage der Bahntrasse sowie der vorgegebenen Bauabschnitte G1 – G11**

Zur Ergebnisauswertung wäre folgendes denkbar:

Statistik der ermittelten Wassermenge (d. h. Minimal-, Maximal-, Mittelwert sowie Standardabweichung), die durch die Randbedingungen abgeflossen ist, um die GW-Absenkungsziele für die Baugruben der geplanten Bahntrasse zu erreichen. Die Registrierung bzw. Abfrage der abgeflossenen Wassermengen erfolgt durch die FEFLOW-Fluxgruppen.

Ergebnisdaten können in tabellarischer Form zur Auswertung ausgegeben werden, z. B. in folgendem Format

| Name | Min | Mittel | Max | Std. | Zeitraum |
|---|---|---|---|---|---|
| Bauabschnitt1 | Qmin1 | Qmittel1 | Qmax1 | std1 | Begin1, Ende1 |
| Bauabschnitt2 | Qmin2 | Qmittel2 | Qmax2 | std2 | Begin2, Ende2 |

…

Mit zu registrierende Ergebnisse sind Wasserstände in den GWM- bzw. Beobachtungspunkten:

| Name | Min | Mittel | Max | Std. | Zeitraum |
|---|---|---|---|---|---|
| GWM1 | Wmin1 | Wmittel1 | Wmax1 | Wstd1 | Beginn, Ende |
| GWM2 | Wmin2 | Wmittel2 | Wmax2 | Wstd2 | Beginn, Ende |

…

Zu registrieren sind neben den P-Werten auch die Elemente, bei denen der Parameter geändert wurde. Dabei sind generell auch Min-, Mittel- und Max-Werte der geänderten Parameter zu registrieren.

Man kann die Ergebnisse der Statistik bzw. Häufigkeit (z. B. als Histogramm) bzgl. der geografischen Lage der Bauabschnitte bzw. GWM räumlich darstellen.

## 1.5 Optimierung der Standorte

Die Optimierung der Standorte von Absenk- bzw. Abfanganlagen erfordert in der Regel nach dem eigentlichen Optimierungsprozess weitere Analysen durch erneute Modellläufe oder Auswertung der Ergebnisdateien (in der Regel vom Typ .dac) der optimalen Lösung. Diese Dateien enthalten auch alle Informationen, die für die weitere Betrachtung und die Ergebnisbewertung notwendig sind. Die Auswertung dieser Dateien kann unter Umständen sehr aufwändig sein. Aus diesem Grund ist als Ergebnis neben den Modell- und Standardergebnisdateien der „optimalen" Lösung eine statistische Ergebnisdarstellung gewünscht. Ebenso hilft eine kartografische Darstellung, die gefundene Lösung zu bewerten. Hierzu ist entweder eine kartografische Darstellung direkt im Portal von Interesse. Ebenso können aber die raumbezogenen Informationen auch unter Verwendung eines Dateiformats weitergegeben werden, das von Geografischen Informationssystemen (GIS) gelesen werden kann, z. B. Shape- oder ASCII-Dateien.

Bei der statistischen Ergebnisdarstellung ist generell das Verhalten der zu optimierende Größe (hier: im Wesentlichen Pumpmenge und Lage der Pumpanlage) und der einzuhaltenden Nebenbedingung (Wasserstände und Konzentrationen an Grundwassermessstellen) wiederzugeben. Es muss dabei zwischen stationären und instationären Betrachtungen unterschieden werden, da im letzteren Fall die zeitliche Abhängigkeit zu berücksichtigen ist.

Statistische Auswertung der optimierten Größe im stationären Fall:

a. Es ist eine Liste der in der optimierten Variante verwendeten Anlagen zu liefern, das sind alle im Rahmen der Optimierung gesetzten Brunnen-Randbedingungsknoten. Der Standort ist jeweils in Form von Koordinaten anzugeben.
b. Für jede Anlage sind die gefundene Pumpmenge und die ggf. daraus resultierenden Betriebskosten anzugeben.
c. Die Gesamtpumpmenge und die Gesamtbetriebskosten sind anzugeben.

Statistische Auswertung der Nebenbedingungen im stationären Fall:

d. Für alle einzuhaltenden Nebenbedingungen sind die erreichte Größe und die Differenz zur einzuhaltenden Größe anzugeben.

Statistische Auswertung der optimierten Größe im instationären Fall:

e. Es ist eine Liste der in der optimierten Variante verwendeten Anlagen zu liefern, das sind alle im Rahmen der Optimierung gesetzten Brunnen-Randbedingungsknoten. Der Standort ist jeweils in Form von Koordinaten anzugeben.

f. Für jede Anlage und im Gesamten sind Minimum, Maximum und Durchschnitt der gefundenen Pumpmengen und die ggf. daraus resultierenden Betriebskosten anzugeben. Da die instationäre Betrachtung nicht frei, sondern nur unter Berücksichtigung definierter Perioden möglich ist, sind für jede Periode die gefundenen anlagenspezifischen Pumpmengen und Betriebskosten und auch die gesamte Menge und Kosten anzugeben.

g. Die Gesamtpumpmenge und die Gesamtbetriebskosten sind anzugeben.

h. Wünschenswert ist auch eine Grafik, die für jede Anlage und im Gesamten die Pumpmenge bzw. Betriebskosten über der Zeit darstellt.

Statistische Auswertung der Nebenbedingungen im instationären Fall:

i. Für alle einzuhaltenden Nebenbedingungen sind für jede Periode die erreichte Größe und die Differenz zur einzuhaltenden Größe anzugeben. Über alle Perioden betrachtet sind jeweils Minimum, Maximum und Durchschnitt anzugeben.

j. Wünschenswert ist auch eine Grafik, die für jede Nebenbedingung das Verhalten der einzuhaltenden Größe (Wasserstand/Konzentration) darstellt..

## 1.6 Zusammenfassung

Aus den Betrachtungen zur Sensitivitätsanalyse lässt sich folgendes ableiten bzw. verallgemeinern: Es sind zwei Typen FEFLOW-Daten als Ergebnisdaten für die statistische Auswertung vorzubereiten:

1. Geänderte bzw. modifizierte Parameter, welche element- bzw. knotenbezogen als Min-, Mittel-, Max-Wert und Standardabweichung erfasst werden sollen. Dabei können die Min-, Mittel-, Max-Werte und die Standardabweichung auch zeitabhängig sein. In diesem Fall sind die Ergebnisse auch in Form entsprechender Zeitreihen verfügbar zu machen.
2. Berechnete Ergebnisse (Wasserstände, Konzentrationen, Temperaturen oder Wassermengen usw.), die ebenfalls element-, knoten- und observationspunktbezogen als Min-, Mittel-, Max-Wert und Standardabweichung erfasst werden sollen. Im instationären Fall sind die Ergebnisse auch in Form entsprechender Zeitreihen verfügbar zu machen.

Die Ergebnisdaten können in tabellarischer Form für die betrachteten Objekte (GWM, Brunnen, bestimmte Randbedingungen etc.) zur Auswertung ausgegeben werden. Eine optionale grafische Darstellung der Statistik bzw. Häufigkeit (z. B. als Histogramm) soll durch Einbeziehen der geografischen Lage der betrachteten Objekte erfolgen.

Die Anforderungen der Standortoptimierung an die Ergebnisdarstellung sind andere. Es sind drei Gruppen von Ergebnisdaten zu unterscheiden:

1. Das optimierte Modell ist als FEM-Datei zurückzuliefern.
2. Tabellarisch sind folgende Listen darzustellen
    a. eingesetzte Anlagen inkl. ihrem Standort
    b. deren eingesetzte Pumpleistungen/entstandenen Betriebskosten
    c. eingehaltene Größe der Nebenbedingungen
    d. bei instationären Betrachtungen sind die Ergebnisse aus b und c als Zeitreihe zuliefern
3. Bei instationären Betrachtungen sollten das zeitliche Verhalten der Anlagen (siehe auch Punkt 2 b) und der Nebenbedingungen der Anlagen (siehe auch Punkt 2 c) grafisch als Diagramm dargestellt werden.

# 2 Softwaredesign der verteilten Variantenrechnung

Aus den Anforderungen der Endanwender an die verteilte Variantenrechnung aus Deliverable 2: Spezifikation und Softwareentwurf der verteilten Variantenrechnung und der Definition der gewünschten bzw. erforderlichen Ergebnisse aus Kapitel 1 lässt sich der Ablauf einer verteilten Variantenrechnung am Beispiel der Sensitivitätsanalyse skizzieren.

## 2.1 Ablauf der Sensitivitätsanalyse

Der Ablauf einer verteilten Variantenrechnung wird im Folgenden anhand der Sensitivitätsanalyse beschrieben. Unter Einbeziehung der Anforderungen aus Anwendersicht (vgl. Deliverable 2) und der hieraus abgeleiteten Ergebnisse und Ergebnisdarstellungen (vgl. Kapitel 1) lässt sich das in Abbildung 2-1 dargestellte UML Aktivitätsdiagramm erstellen. Es verdeutlicht den Ablauf einer Sensitivitätsanalyse aus Anwendersicht.

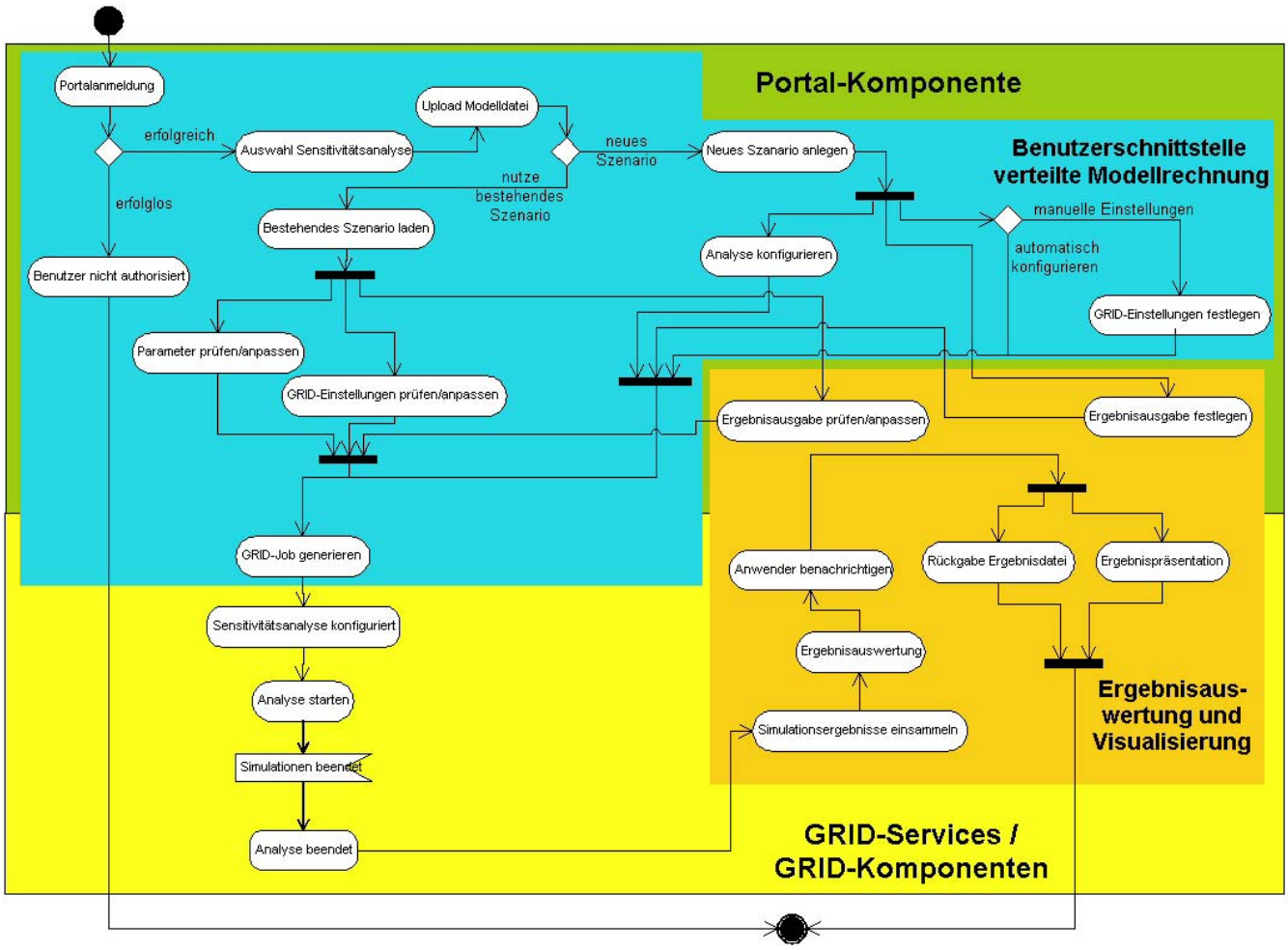


**Abbildung 2-1: UML-Aktivitätsdiagramm der Sensitivitätsanalyse**

In dem Aktivitätsdiagramm sind logisch zusammengehörige Blöcke farblich gekennzeichnet. Hierbei wurde zwischen zwei Hauptblöcken unterschieden. Abläufe, die Benutzerinteraktion benötigen und daher innerhalb der Portalkomponente anzuordnen sind, wurden durch den grünen Block hinterlegt, während automatisiert ablaufende Prozesse als gelb hinterlegte GRID Komponenten dargestellt sind.

Innerhalb der zwei großen Blöcke wurde eine weitere farblich gekennzeichnete Unterteilung vorgenommen. Die blau hinterlegte Benutzerschnittstelle zur verteilten Modellrechnung wurde in Deliverable 2 spezifiziert. Der orange markierte Block der Ergebnisauswertung und Visualisierung wird im Folgenden näher beschrieben. Nicht Bestandteil des UML-Aktivitätsdiagrammes ist der Ablauf, der aus dem GRID Job der Sensitivitätsanalyse die einzelnen Simulationsjobs erzeugt und diese in der GRID-Umgebung ausführt. Dieser Ablauf wird in Bezug auf die Ergebnisauswertung und Visualisierung als Black Box verstanden und ist in Abbildung 2-1 lediglich als ein Warten auf das Signal „Simulation beendet“ enthalten.

## 2.2 Komponentenstruktur der verteilten Variantenrechnung

Aus dem Ablauf der verteilten Variantenrechnung (z. B. der Sensitivitätsanalyse) lassen sich die erforderlichen Funktionalitäten ableiten und zusammenhängenden Komponenten zuordnen. Bei der Zuordnung der Funktionalität ist zusätzlich jedoch auch die Struktur eines GRIDs zu berücksichtigen. Eine schematische GRID-Übersicht zur verteilten Variantenrechnung mittels FEFLOW ist in Abbildung 2-2 dargestellt.

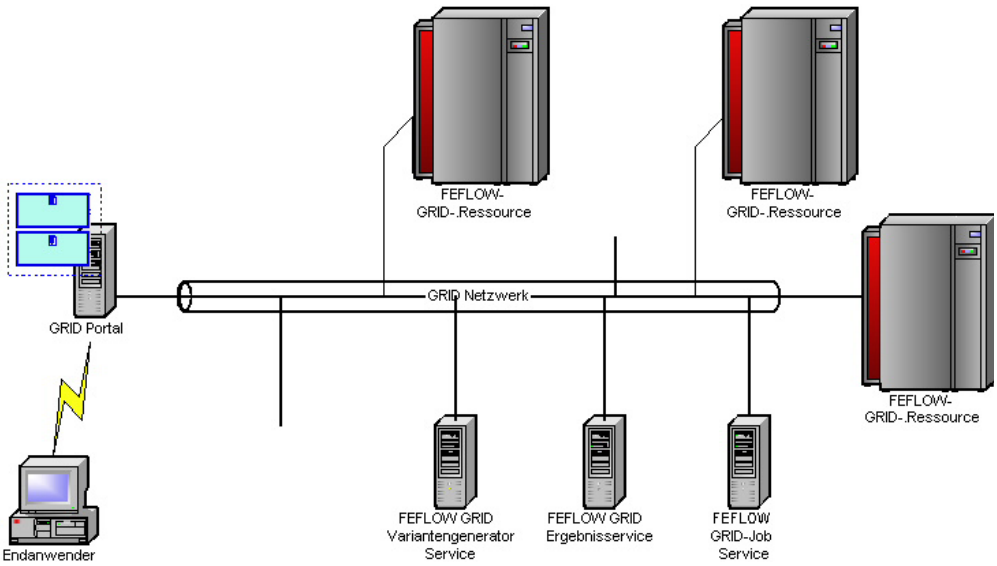


**Abbildung 2-2: Schematischer GRID-Aufbau zur Variantenrechnung**

In einer GRID Infrastruktur ist dabei besonders zu beachten, dass Komponenten auf unterschiedlichen Ressourcen angesiedelt sein können. Eine Gliederung der Komponenten in Form abgeschlossener Dienste mit definierten Schnittstellen ist daher sinnvoll.

Für die verteilte Variantenrechnung wurden fünf wesentliche Komponenten identifiziert, die jeweils als separater GRID Service bzw. GRID Komponente realisiert werden:

1. GRID Portal

   Das GRID Portal dient als Einstiegspunkt für den Endanwender. Neben den Oberflächen zur allgemeinen Verwaltung (z. B. Benutzerinformationen), die vom Portalframework bzw. externen Entwicklungen zur Verfügung gestellt werden, sind anwendungsspezifische Konfigurationsmöglichkeiten der verteilten Variantengenerierung zu entwickeln (z. B. Oberflächen zur Jobkonfiguration; vgl. Deliverable 2). Zu klären wäre noch die Zugehörigkeit vordefinierter Varianten bzw. Ergebnisauswertungen. Diese könnten logisch entweder Bestandteil der Serviceimplementierungen (Variantengenerator Service bzw. Ergebnisservice) sein, aber auch als vordefinierte Benutzereinstellungen in die Portalkomponente integriert werden.

   Anhand der vom Endanwender getroffenen GRID Jobdefinitionen wird aus dem Portal heraus ein FEFLOW GRID Job Service kontaktiert, der für die Verwaltung und Abarbeitung des FEFLOW GRID Jobs zuständig ist.

2. FEFLOW GRID Job Service

   Der FEFLOW GRID Job Service ist für die Verwaltung abgeschlossener Aufgaben (z. B. eine Sensitivitätsanalyse oder eine Optimierung) zuständig. Er beinhaltet die Modelldatei, Informationen über die zu belegenden/variierenden Parameter, die Form der Ergebnisrückgabe und optionale Zusatzinformationen, bspw. Shapedateien zur Eingrenzung des zu variierenden Parametergebiets. Bestandteil des GRID Jobs sind darüber hinaus die Python-Skripte zur Variantengenerierung und Ergebnisauswertung. Vordefinierte Skripte könnten ggf. auch vereinfacht übergeben werden, z. B. in Form der Verfahrensbezeichnung, um den Aufruf durch externe Service-Implementierungen zu erleichtern.

   Der GRID Job Service erzeugt aus den Job-Informationen die gewünschte bzw. erforderliche Zahl an Simulationsjobs. Dies erfolgt unter Zuhilfenahme des generischen Variantengenerator Service, der modellunabhängig die Parametrisierung der jeweiligen Varianten erzeugt. Diese Parametrisierungen werden durch den GRID Job Service zusammen mit der Modelldatei und dem Rückgabeskript zu einem vollständigen und unabhängigen Simulationsjob zusammengesetzt, der dann an eine FEFLOW GRID Ressource zur Ausführung übergeben wird.

   Nach der Ausführung der einzelnen Simulationen werden die Ergebnisse der Einzelsimulationen mittels des generischen FEFLOW GRID Ergebnisservice zu dem spezifizierten Gesamtergebnis zusammengestellt.

3. FEFLOW GRID Ressource

   Die FEFLOW GRID Ressource ist zur Ausführung eines einzelnen Simulationsjobs auf der lokalen Ressource (Knoten, Cluster) zuständig. Hierzu bekommt die GRID Ressource das vollständige Python-Skript übergeben. Es besteht aus dem Simulationscode zum Start und zur Ausführung der Simulation sowie dem Skript zur Rückgabe der Ergebnisse. Es beinhaltet ebenfalls die kodierte Modelldatei und optionale Daten zur Auswahl der zu belegenden Parameter. Dieses Skript kann anschließend direkt auf dem aktuellen Knoten ausgeführt werden oder es wird an den lokalen Scheduler übergeben. Nach der Ausführung des Jobs wird der aufrufende Service (FEFLOW GRID Job Service) benachrichtigt.

4. FEFLOW GRID Variantenservice

   Der Variantenservice erzeugt aus dem gewählten Standardverfahren bzw. dem benutzerdefinierten Skript die einzelnen Simulationsjobs. Hierbei soll der Service generisch arbeiten, also eine Parametrisierung der Variante er-

möglichen, ohne Wissen über das darunter liegende Modell oder den gewählten Parameter zu benötigen. Ziel dabei ist es, den Variantengenerator flexibel zu halten, sodass er durch spezielle Implementierungen bspw. einen Optimierungsalgorithmus ersetzt werden kann.

In Bezug auf den Einsatz eines Optimierungsalgorithmus ist zu beachten, dass es möglich sein muss, den Variantengenerator iterativ zu betreiben, da eine modellbasierte Optimierung typischerweise ein iterativer Prozess mit einem definierten Abbruchkriterium ist. Daher muss die Möglichkeit berücksichtigt werden, dass Ergebnisse bisheriger Simulationen in der Regel wieder in die Erstellung neuer Varianten einfließen. Es muss also eine Interaktion zwischen Varianten- und Ergebnisservice ermöglicht werden. Diese Möglichkeit wird im Rahmen des Deliverable 4: Spezifikation und Softwareentwurf der Integration der verteilten Optimierung näher spezifiziert werden.

5. FEFLOW GRID Ergebnisservice

   Analog zur Variantengenerierung arbeitet der FEFLOW GRID Ergebnisservice ebenfalls als generische Variante. Er erzeugt wiederum modell- und parameterunabhängig aus den Ergebnissen der Einzelsimulationen das gewünschte Gesamtergebnis. Auch der Ergebnisservice sollte so flexibel gestaltet werden, dass er durch andere Verfahren ersetzt werden kann.

## 2.3 Softwareentwurf der Ergebnisauswertung und Visualisierung

Die Ergebnisauswertung und Visualisierung erfolgt analog der Variantengenerierung, d. h. sie ist skriptbasiert. Die Ergebnisse der einzelnen Simulationen werden mittels Python-Skript aus dem FEFLOW-Modell ausgelesen und auf *stdout* ausgegeben (*stderr* ist für die Ausgabe von Nachrichten bzw. Fehlermeldungen vorgesehen). Das Format der Ausgabe ist frei definierbar. Ergebnisse könnten sowohl im ASCII-Format oder aber als base64 kodierte Binärdaten ausgegeben werden. Zur Ausgabe mehrerer unterschiedlicher Parameter bietet sich darüber hinaus die Verwendung einer Multifile-Struktur an. Das genaue Format wird im Rahmen der Entwicklung der Ergebniskomponente entsprechend den auftretenden Anforderungen angepasst. Da sowohl das Erzeugen des Datenstroms als auch dessen Auswertung mittels Python-Skript realisiert werden kann, ist eine nachträgliche Anpassung oder Erweiterung jederzeit unproblematisch möglich. Bei der Ergebnisrückgabe ist zu beachten, dass die Belegung des zu variierenden Parameters in der Regel Bestandteil der Ergebnisrückgabe sein wird. Dies ist erforderlich, da die zu setzenden Parameterwerte evt. unter Zuhilfenahme von Zufallswerten direkt auf der GRID Ressource berechnet werden und daher dem FEFLOW GRID-Job Service nicht bekannt sind, jedoch zur Ergebnisauswertung oder als Ergebnis für den Anwender benötigt werden.

Die Ergebnisse der Einzelsimulationen, die auf den FEFLOW GRID Ressourcen erzeugt werden, können vom FEFLOW GRID Job Service abgerufen und an den FEFLOW Ergebnisservice zur Auswertung übergeben werden. Durch den Ergebnisservice werden die Einzelergebnisse dann zu dem gewünschten Gesamtergebnis (vgl. Kapitel 1) zusammengestellt.

Für die Ergebnispräsentation ist eine Darstellung der Daten im Portal zu entwickeln. Hierbei wird in einem ersten Schritt die Präsentation der Ergebnisse in tabellarischer Form realisiert werden (vgl. Kapitel 1.6). Die Realisierung einer optionalen grafischen Darstellung insbesondere unter Berücksichtigung der geografischen Lage wird zunächst zurückgestellt. Für diese Darstellung ist insbesondere zu prüfen, ob eine Realisierung innerhalb des Portals sinnvoll ist oder stattdessen eine Rückgabe der Ergebnisse an den Anwender erfolgt, um eine Visualisierung und Auswertung der Daten mittels eines geografischen Informationssystems oder FEFLOW zu ermöglichen.

## Anhang A: Python-Skript zur Generierung eines einzelnen FEFLOW-Jobs

Das nachfolgende Python-Skript ist ein Beispiel zur Erzeugung einer einzelnen FEFLOW Simulation in Skriptform. Ein vergleichbarer Mechanismus kommt im FEFLOW GRID Job Service zum Einsatz, um aus der Variantendefinition und einer Modelldatei ein Jobskript zu erzeugen.

```
# Samplescript to create a single FEFLOW Job-Script

# import of required Python packages

import sys
import os
import StringIO

# encode FEM-file using base64 coding
def encodeFemFile(femFile, resultFile) :
 femFile.seek(0)
 fTemp = StringIO.StringIO()
 fTemp.write(femFile.read().encode("base64"))
 fTemp.seek(0)
 lList = fTemp.readlines()
 fTemp.close()
 for i in range(0, len(lList)) :
  s = lList[i]
  if s[len(s)-2:len(s)] == "\r\n" :
    s = s[0:len(s)-2]
  elif s[len(s)-1:len(s)] == "\n" :
    s = s[0:len(s)-1]
  resultFile.write(s)
 resultFile.write(os.linesep)

# define mutifile separator string
separator = "--FEFLOWGridJob"

# open files:
#   1. input file script template
#   2. input FEM-file
#   3. output job scriptfile
fInScript = open(sys.argv[1], "r")
fInFem = open(sys.argv[2], "rb")
```

```
fOut = open(sys.argv[3], "wb")

# create output file as multifile of at least script code
# and base64 encoded FEM-file
fOut.write(separator + os.linesep)
fOut.write(fInScript.read())
fOut.write(separator + os.linesep)
encodeFemFile(fInFem, fOut)
fOut.write(separator + "--" + os.linesep)
```

## Anhang B: Template-Skript zur Ausführung eines FEFLOW Skript-Jobs

Das folgende Python-Skript beschreibt ein Template-Skript zur Ausführung einer FEFLOW Simulation. Das Skript bildet die Basis für den Codeblock des FEFLOW Jobs und extrahiert aus dem Mutilfileskript die FEM-Datei, dekodiert diese und führt die Simulation durch. Die Stellen, an denen der Skriptcode zur Parametermodifikation und zur Ergebnisrückgabe einzufügen ist, sind entsprechend markiert.

```
# import of required Python packages
import sys
import os
import tempfile
import multifile

# definition of the mutilfile separator string
__FeflowResultSeparator = "--FEFLOWGridJobResult"

# routine to decode the second muiltfile part (FEM-file)
def readFemFile() :
 __FeflowJobFile.next()
 return __FeflowJobFile.read().decode("base64")

# create temporary FEM-file from base64 encoded data
lTmpFile = tempfile.mktemp(".fem")
sys.stderr.write("Storing FEF-File... " + lTmpFile + os.linesep )
open(lTmpFile, "wb").write(readFemFile())

# startup FEFLOW and load FEM-file
sys.stderr.write("Running Feflow..." + os.linesep )
import ifm
doc = ifm.loadDocument(lTmpFile)

### add modification of parameters here

# run simulation
doc.startSimulator()

### return simulation results here

# stop simulation, remove temporary FEM-file and  shutdown FEFLOW
doc.stopSimulator()
```

```
os.unlink(lTmpFile)
sys.stderr.write("finished, exit... " + os.linesep )
ifm.shutdownKernel()
```

## Anhang C: Beispielhafter FEFLOW-Job einer einzelnen Simulation

Das nachfolgende Python-Skript stellt einen exemplarischen FEFLOW-Job in Skriptform dar. Das Skript enthält den Ausführungscode und die base64 kodierte Modelldatei in Form eines Mutifiles.

```
--FEFLOWGridJob
# import of required Python packages
import sys
import os
import tempfile
import multifile

# definition of the mutilfile separator string
__FeflowResultSeparator = "--FEFLOWGridJobResult"

# routine to decode the second muiltfile part (FEM-file)
def readFemFile() :
 __FeflowJobFile.next()
 return __FeflowJobFile.read().decode("base64")

# FEFLOW callbacks to demonstrate the handling of timestep based
# data
def preTimeStep(doc) : sys.stderr.write("Pre-TS: " + str(doc.getAbsoluteSimulationTime()) + os.linesep )
def postTimeStep(doc) : sys.stderr.write("...TS: done" + os.linesep )

# create temporary FEM-file fom base64 encoded data
lTmpFile = tempfile.mktemp(".fem")
sys.stderr.write("Storing FEF-File... " + lTmpFile + os.linesep )
open(lTmpFile, "wb").write(readFemFile())

# startup FEFLOW, load FEM-file
sys.stderr.write("Running Feflow..." + os.linesep )
import ifm
doc = ifm.loadDocument(lTmpFile)

# modify material flow storativity using random values from [0, 10]
import random
for i in range(0, doc.getNumberOfElements()) :
 doc.setMatFlowStorativity(i, 10 * random.random())
```

```
# run simulation
doc.startSimulator()

# sample routine to create the result output (hydraulic head)
# of each FEM-node
# create output as mutilfile
print __FeflowResultSeparator
print “FlowHeadValue:"
for i in range(0, doc.getNumberOfNodes()) :
 print str(i + 1) + " " + str(doc.getResultsFlowHeadValue(i))
print __FeflowResultSeparator

# shutdown FEFLOW
doc.stopSimulator()
os.unlink(lTmpFile)
sys.stderr.write("finished, exit... " + os.linesep )
ifm.shutdownKernel()

--FEFLOWGridJob
UFJPQkxFTTogUFJPQkxFTTogQSBuZXcgZmVmbG93IH
….
AAAAAAACCAAAAEAAAmrA==
--FEFLOWGridJob—
```