TINF13IN GmbH

Pflichtenheft für

Software-Engineering Gruppenarbeit

erstellt von:
Alexander Rezmer
Fabian Zeiler
Christian Gmeiner

30. Oktober 2014
Version: v0.9

Inhaltsverzeichnis

1 Zielbestimmung **4**
- 1.1 Musskriterien . . . 4
- 1.2 Wunschkriterien . . . 4
- 1.3 Abgrenzungskriterien . . . 5

2 Produkteinsatz **6**
- 2.1 Anwendungsbereiche . . . 6
- 2.2 Zielgruppen . . . 6
- 2.3 Betriebsbedingungen . . . 6

3 Produktübersicht **8**

4 Produktfunktionen **9**

5 Produktdaten **11**

6 Produktleistungen **12**

7 Qualitätsanforderungen **13**

8 Benutzungsoberfläche **15**

9 Technische Produktumgebung **16**
- 9.1 Software . . . 16
- 9.2 Hardware . . . 16
- 9.3 Orgware . . . 17
- 9.4 Produkt-Schnittstellen . . . 17

10 Spezielle Anforderungen an die Entwicklungs-Umgebung **18**
- 10.1 Software . . . 18
- 10.2 Hardware . . . 18
- 10.3 Orgware . . . 18
- 10.4 Produkt-Schnittstellen . . . 18

11 Gliederung in Teilprodukte **19**

12 Anhang **20**
- .1 Begriffsdefinitionen . . . 20

.2 Abkürzungsverzeichnis 20
.3 Abbildungsverzeichnis 20
.4 Aufwandsabschätzung 20
.5 Datenflussdiagramme 20
.6 Tabellenverzeichnis 20

1 Zielbestimmung

Das im Folgenden beschriebene Programm soll die Grundlage für ein Bewertungssystem für Prüfer sein. Mit diesem System soll eine gerechte und einfache Bewertung ermöglicht werden. Dies wird sichergestellt indem die einzelnen Bewertungen auf einen gemeinsamen Score umgerechnet werden. Durch verschiedene Gewichtungen der Scores ergibt sich die Möglichkeit einzelne Bewertungen stärker zu werten als andere. Eine weitere Funktionalität des Programmes soll es dem Prüfer ermöglichen zum Score ebenfalls eine Rückmeldung anzuhängen um dem Prüfling ein Feedback zu geben. Genau so soll der Prüfling (ggf. Student) ein Feedback anfordern können um eine Begründung zu für seine Bewertung erhalten zu können.

1.1 Musskriterien

Die Software muss eine korrekte Umrechnung der Bewertungen in gültige Scores beherrschen, welche die Prüfer eintragen. Da es sich bei diesen um teilweise sensible Daten handelt, müssen diese gut geschützt sein, sodass sie von außerhalb nicht verändert werden können. Dieses Programm bietet die Möglichkeit durch ein zweistufiges holistisches Bewertungssystem die Bewertungen einzutragen. Bei diesem zweistufigen System kann der Prüfer die Stufen selbst so gewichten wie er es für sinnvoll hält. Mehrere Bewertungen können miteinander verrechnet und einzeln gewichtet werden. Am Ende soll der Prüfer ein Score bekommen, welcher sich aus seinen Bewertungen und Gewichtungen errechnen lässt. Die Software muss auf Windows PCs lauffähig sein, da dies die am meisten verwendete Laufzeitumgebung ist. Des weiteren muss das System einfach zu bedienen sein. Der Prüfer muss die Ergebnis seiner Bewertung einfach an die Studenten weiter geben können. Dies kann durch einen Ausdruck dieser statt finden wie auch durch ein eigenes Programm welches den Studenten Zugriff auf die jeweiligen Scores liefert. Umrechnung von der Inversen muss auch voll funktionsfähig sein.

1.2 Wunschkriterien

Das Grundprogramm soll wie ein Gerüst dienen, sodass man verschiedene Add-On einbinden kann. Dieses Zusatzapplikationen können weitere Bewertungsmöglichkeiten beziehungsweise Umrechnungen zwischen diesen Bewertungen sein. Es kann sich dabei auch um eine Anbindung an bestehende Systeme verschiedener anderer Anbieter handeln. Später, wenn das Backend funktionsfähig ist, soll durch eine Website ein Zugriff ermöglicht werden um von Mobilen Endgeräte oder jedem anderen Gerät welches einen Browser besitzt darauf zugreifen zu können. Des weiteren soll eine Application Programming

Interface (Programmierschnittstelle) (API) implementiert werden, mit welcher andere Programme mit diesem Programm kommunizieren können. Es soll eine iOS App erstellt werden um bequem von iPad/iPhone darauf zugreifen zu können.

1.3 Abgrenzungskriterien

Das Produkt soll keine Software werden, mit welcher die Studenten untereinander verglichen werden. Dies kann gegebenen falls durch Add-On ermöglicht werden, es gehört jedoch nicht zu diesem Projekt dazu. Des weiteren soll die Software kein Netzwerk darbieten in welchem unterschiedlichen Prüfer sich mit anderen austauschen können und die Prüfungsergebnisse untereinander vergleichen zu können.

2 Produkteinsatz

Der geplante Einsatz des Systems ist die Grundlage für Benutzungsoberfläche und Qualitätsanforderungen.

2.1 Anwendungsbereiche

Das System soll vor allem zur Bewertung von Schülern und Studenten dienen. Es können jedoch auch andere Institutionen oder Firmen, welche ein einheitliches Bewertungssystem wollen, welches eine faire Bewertung ermöglicht. Es muss jedoch garantiert werden, dass keine fremden Personen Zugriff auf das System bekommen, da es sich bei solchen Bewertungen um teils sensible Informationen handelt. Studenten sollten die einfache Möglichkeit besitzen nachdem sie einen Test durchlaufen haben ihre Bewertung zu erfahren und diese auf einer einfachen Oberfläche teils grafisch dargestellt werden soll um einen Vergleich zu ermöglichen.

2.2 Zielgruppen

- Prüfer:
  Der Prüfer ist der Ersteller sowie der Bearbeiter der Prüfung. Dieser muss sich sinnvolle Bewertungskriterien überlegen, sowie in das System eintragen. Der Prüfer benötigt alle Funktionen zum Eintragen der Bewertungen bzw. der Scores.

- Hochschulen:
  Das System soll transparent für verschiedene Hochschulen verwendet werden. Die Bewertungskriterien sind einfach anpassbar an die jeweiligen Hochschulverordnungen. Zusätzlich sind individuelle Bewertungskriterien stets möglich.

- Verwaltung:
  Die Verwaltung benötigt Zugriff auf das System um die Score auslesen zu können. Diese werden in einem Archiv abgespeichert und für 10 Jahre hinterlegt.

2.3 Betriebsbedingungen

- Physikalische Umgebung:
  Die Software soll später auf jedem Computer lauffähig sein. Die Datenbank mit API wird auf einem Server installiert. Somit besteht die Möglichkeit, dass mehrere Personen gleichzeitig Zugriff auf die Daten haben. Über ein kleines Programm kann der Prüfer sich mit der Datenbank verbinden und darauf zuzugreifen.

- Tägliche Betriebszeit:
  Der Server auf dem die Daten hinterlegt sind muss immer verfügbar sein, damit die Prüfer wie auch die Prüflinge jederzeit mit der Anwendung Zugriff haben.

- Betrieb:
  Es ist ein unbeaufsichtigter Betrieb vorgesehen. Sobald der Server einmal installiert ist, sollte das System ohne weitere Konfigurationen funktionieren. Der Prüfer sollte sich nicht mit der Konfiguration des System belasten müssten.

3 Produktübersicht

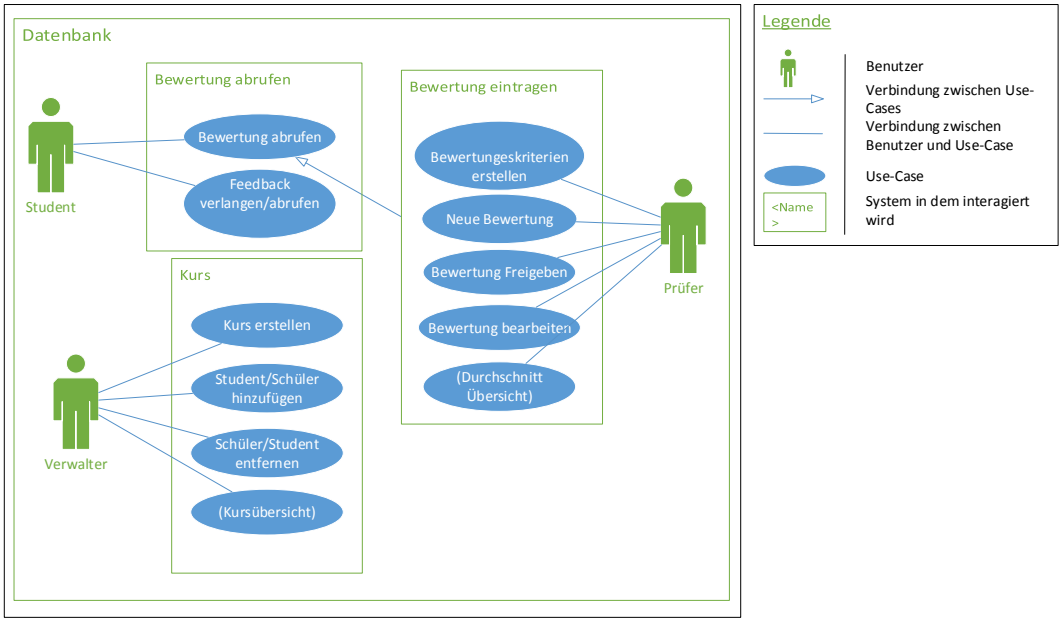

Abbildung 3.1: Use-Case Szenario

4 Produktfunktionen

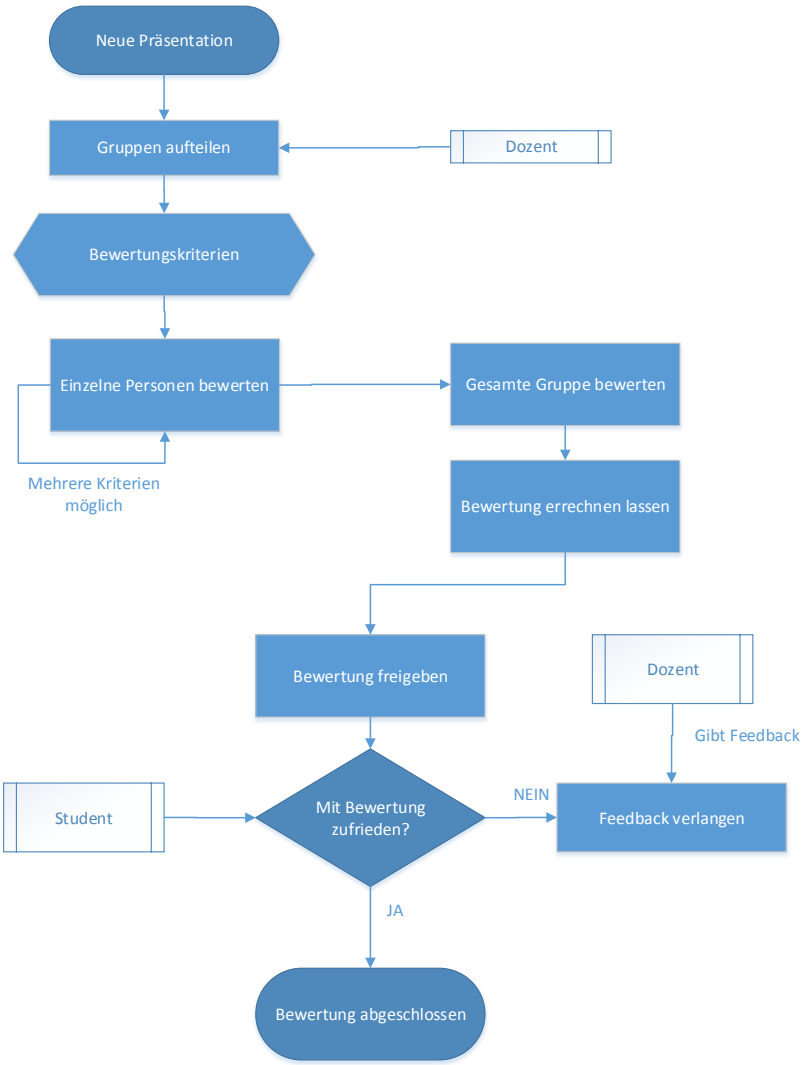

Abbildung 4.1: Beispielprozess einer Bewertung

5 Produktdaten

In den Produktdaten sollten enthalten sein:

- Name des Studenten
- Vorname des Studenten
- Matrikelnummer des Studenten
- Die „Endscores“ des Studenten
- „Zwischenscores“
- Fach vom Score

Insgesamt können von jedem Studenten mehrere Score eingetragen werden. Name, Vorname und Matrikelnummer sind nur einmal in der Datenbank hinterlegt um Redundanz zu verhindern.

6 Produktleistungen

Die Software soll eine Stabilität enthalten, wodurch mehrere Studenten/Dozenten auf einmal darauf zugreifen können.

7 Qualitätsanforderungen

Tabelle 7.1: Qualitätsanforderungen

Produktqualität	sehr gut	gut	normal	irrelevant
Funktionalität				
Angemessenheit			×	
Richtigkeit		×		
Interoperabilität			×	
Ordnungsmäßigkeit		×		
Sicherheit		×		
Zuverlässigkeit				
Reife		×		
Fehlertolleranz			×	
Wiederherstellbarkeit			×	
Benutzbarkeit				
Verständlichkeit	×			
Erlernbarkeit	×			
Bedienbarkeit	×			
Effizienz				
Zeitverhalten				×
Verbrauchsverhalten			×	
Änderbarkeit				
Analysierbarkeit			×	
Modifizierbarkeit				×
Stabilität			×	
Prüfbarkeit			×	
Übertragbarkeit				
Anpassbarkeit				×
Installierbarkeit				×
Konformität			×	
Austauschbarkeit				×

- Funktionalität:
  In der Rubrik Funktionalität spielt vor allem die Sicherheit eine große Rolle, weshalb die Qualitätsanforderungen hier sehr gut sein müssen. Es muss garantiert werden, dass keine Daten geklaut oder missbraucht werden können. Deshalb ist es

ratsam hier mehr Zeit zu investieren. Die Punkte Richtigkeit und Ordnungsmäßigkeit müssen qualitativ nicht so hochwertig sein wie die Sicherheit, weil kleine Pannen bei der Benutzung keine gravierenden Schäden anrichten. Normale Qualität genügt bei Angemessenheit und Interoperabilität.

- Zuverlässigkeit:
  Insgesamt sollte das Programm zuverlässig laufen, jedoch müssen nicht überwiegend Ressourcen dafür verbraucht werden. Sollte das System abstürzen oder nicht ordnungsgemäß laufen, kann man nach einem Neustart weiterarbeiten.

- Benutzbarkeit:
  Viel Augenmerk wird auf die Benutzbarkeit gelegt. Das Programm sollte für den Anwender einfach zu bedienen sein und sich selbst erklären, damit keine Schulung mehr notwendig ist.

- Effizienz:
  Ausschlaggebend ist die Effizienz nicht, da das Bewertungssystem nicht besonders schnell die Score abrufen und verarbeiten muss.

- Änderbarkeit: Für die Zukunft können weitere Features eingebaut werden, wenn man möchte. Grundsteine für weitere Features werden nicht gezielt gelegt.

- Übertragbarkeit: Die Übertragbarkeit spielt eine untergeordnete Rolle, da man nur drauf zugreifen muss. Installieren oder ähnliches muss nicht getan werden.

8 Benutzungsoberfläche

- Intuitive Bedienung
- Selbsterklärend
- Ansprechendes Design

9 Technische Produktumgebung

Auf dem Server wird eine CouchDB-Datenbank verwendet. Auf der Clientseite wird ein Java–Programm installiert. Da wir in der Vergangenheit sehr gute Ergebnisse mit Java erzielt haben und eine CouchDB-Datenbank für dieses Projekt sehr sinnvoll erscheint, da es mit der Datenmenge gut zurrecht kommt und noch performant genug ist.

9.1 Software

- Betriebssystem : Windows 7, Windows 8, Windows 8.1, Mac OS X
- Laufzeitsystem: Als Laufzeitsystem wird ein Java Runtime Environement (JRE) benötigt, welches für die meisten Betriebssysteme von Oracle zur Verfügung gestellt wird. [1]
- Datenbank: Apache CouchDB[2]
  Apache CouchDB ist ein NoSQL Datenbanksystem von Apache, welche mittels JavaScript Object Notation (JSON) die einzelnen Dokumente hinterlegt. Als API stellt CouchDB Hypertext Transfer Protocol (HTTP) zur Verfügung. Daher ist CouchDB speziell für Anwendungen entwickelt, welche mit dem Internet verbunden sind und über dieses kommunizieren. CouchDB kann einfach mittels den von Apache bereitgestellten Paketen auf jedem System installiert werden. Es wird Windows, Linux wie auch Mac OSX unterstützt.
- Client: Java Programm Als Clientsoftware wird ein Java-Programm benötigt. Es wird Java verwendet, da es auf den meisten Geräten läuft.

9.2 Hardware

- Server:
  Die Hardware des Servers muss performant genug sein, um die CouchDB lauffähig auf dem System zu halten. Es wird mindestens ein 32-Bit Betriebssystem benötigt.
- Client:
  Die Mindestanforderungen für die Clients sind ein lauffähiges JRE, welches auf den meisten Betriebssystemen von Oracle zur Verfügung gestellt wird.

[1] http://www.oracle.com/technetwork/java/index.html
[2] http://couchdb.apache.org

9.3 Orgware

Der Server muss mit dem Internet verbunden sein. Es wird empfohlen dies über eine synchrone Standleitung (10TBit/s) zu realisieren, um immer verfügbar zu sein.
Die Clients müssen ebenfalls mit dem Internet verbunden sein. Es wird mindestens 2Mbit/s empfohlen.

9.4 Produkt-Schnittstellen

Anbindung von Java und CoucDB über JSON.

10 Spezielle Anforderungen an die Entwicklungs-Umgebung

10.1 Software

Als Software für die Entwicklung wird Windows 7 und Mac OSX 10.10 verwendet, da diese Systeme zur Verfügung stehen. Des weiteren wird Eclipse als IDE für Java verwendet. Um die Daten abzulegen wird eine CouchDB Datenbank verwendet.

10.2 Hardware

Windows 7

MacBook Pro (13 Zoll, Mitte 2009)
Prozessor 2,26GHz Intel Core 2 Duo
Speicher 4GB 1067 MHz DDR3
Betriebsystem OSX 10.Yosemite

10.3 Orgware

Internet Zugang

10.4 Produkt-Schnittstellen

Internetanbindung von Java und CouchDB Datenbank.

11 Gliederung in Teilprodukte

Drei Teilprodukte

- Logik
  - Zweistufige holistische Bewertung (H2)
  - Rate-to-Score (R2S)
  - Score-to-Rate (S2R)
- GUI
- Backend (Datenbankanbindung inkl. Benutzerauthentifizierung)

12 Anhang

.1 Begriffsdefinitionen

.2 Abkürzungsverzeichnis

API Application Programming Interface (Programmierschnittstelle)

Add-On Erweiterungspaket

JRE Java Runtime Environement

JSON JavaScript Object Notation

HTTP Hypertext Transfer Protocol

.3 Abbildungsverzeichnis

3.1 Use-Case Szenario 8

4.1 Beispielprozess einer Bewertung 10

.1 Datenfluss Bewerter-Student 22
.2 Datenfluss Java-Datenbank 22
.3 Datenfluss Umrechnung S2R, H2, R2S 23

.4 Aufwandsabschätzung

.5 Datenflussdiagramme

.6 Tabellenverzeichnis

7.1 Qualitätsanforderungen 13

	#	Kategorie	Klasse	Anzahl		Gewichtung		Beitrag
	1	Eingabedaten	einfach		x	3	=	0
			mittel	3	x	4	=	12
			komplex		x	6	=	0
	2	Abfragen	einfach		x	3	=	0
			mittel		x	4	=	0
			komplex	2	x	6	=	12
	3	Ausgaben	einfach		x	4	=	0
			mittel	4	x	5	=	20
			komplex		x	7	=	0
	4	Datenbestände	einfach		x	7	=	0
			mittel	2	x	10	=	20
			komplex		x	15	=	0
	5	Referenzdaten	einfach		x	5	=	0
			mittel		x	7	=	0
			komplex	3	x	10	=	30
uFP		**unbewertete *Function Points***					=	**94**

		Einflussfaktor mit Skala	<	Gewichtung
	0	Grundstock		70
	1	Verflechtung mit anderen Anwendungssystemen	5	3,0
	2	Dezentrale Daten, dezentrale Verarbeitung	5	2,5
	3	Transaktionsrate	5	4,0
Verarbeitungslogik	4a	Rechenoperationen	10	8,1
	4b	Kontrollverfahren	5	5,0
	4c	Ausnahmeregelungen	10	6,0
	4d	Logik	5	4,0
	5	Wiederverwendbarkeit	5	3,5
	6	Datenbestandskonvertierungen	5	3,0
	7	Anpassbarkeit	5	2,0

E	**Einflusskoeffizient**	**0,01**	**x**	**111,1**	**=**	**1,11**	
FP	**Bewertete Function Points**	**E**	**x**	**uFP**	**=**	**104,43**	
PM	**Umrechnung anhand der IBM-Tabelle**				**=**	**8,27**	*P-Monate*
Dauer	**Umrechnung nach Boehm's Formel für ...**			Dialog	**=**	**5,24**	*K-Monate*

S = Stapelsystme ; D = Dialogsysteme ; E = Echtzeitsysteme

Größe	**Umrechnung durch Division von PM durch Dauer**	**=**	**1,58** *Mitarbeiter*

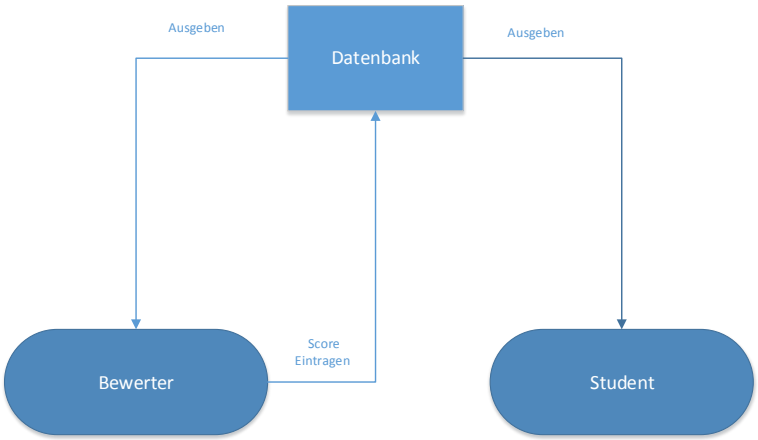

Abbildung .1: Datenfluss Bewerter-Student

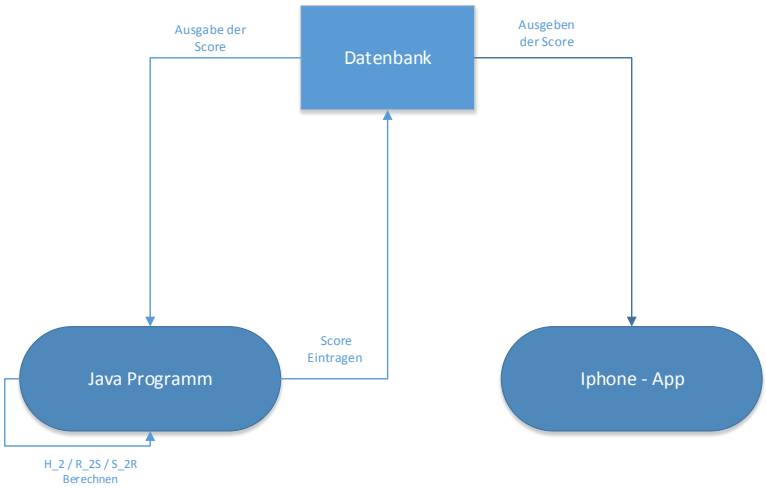

Abbildung .2: Datenfluss Java-Datenbank

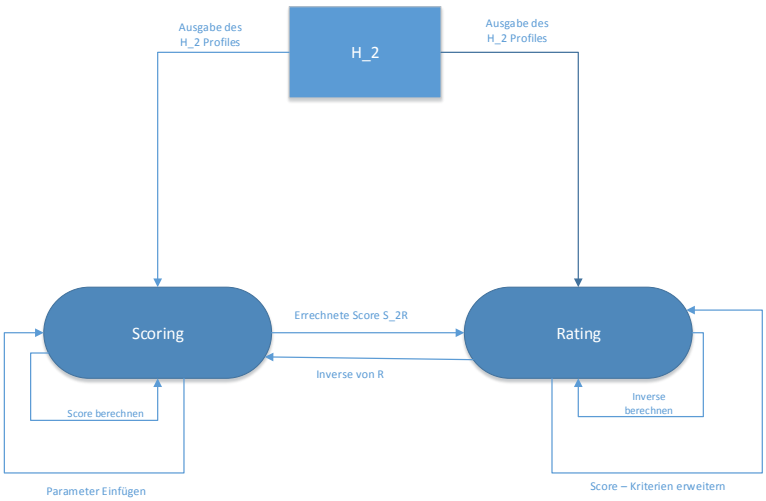

Abbildung .3: Datenfluss Umrechnung S2R, H2, R2S