Berlin – 02.01.2024

Projekt Super Superhirn

Gruppe #13

B42 Software Engineering 2

- Till Giesa (584373)
- Louis Beul (588411)
- Marc Sabrautzki (584673)
- Ole Gleß (564153)
- Philipp Röber (584672)

Hochschule für Technik und Wirtschaft Berlin

University of Applied Sciences

Agenda

1. **Software-Architektur**
   a. Überlegungen zur Architektur
   b. Architektur im Wandel
   c. Feinentwurf und Integration der Netzwerkanbindung
2. **Erfüllung der nicht-funktionalen Anforderungen**
   a. Anforderungen im Detail
   b. Maßnahmen zur Validierung
3. **Qualitätssicherungsmaßnahmen**
   a. Ping-Pong Programming
   b. Code Reviews
   c. Testautomatisierung & CI-Pipeline
4. **Reflexion – Optimierungsmöglichkeiten**
5. **Screenshots**

1. Software-Architektur

Hochschule für Technik
und Wirtschaft Berlin
University of Applied Sciences

Überlegungen zur Architektur

- **Logische Zerlegung** in die Module View, Logic und Data
  - **View** – realisiert die Schnittstelle zum Benutzer
  - **Logic** – beherbergt die Spiellogik
  - **Data** – organisiert und speichert die Spielstände
- Kunde deutete bereits früh auf **Erweiterungen** hin
- Fokus deshalb auf **Erweiterbarkeit** und **Modularität** gesetzt
- Mögliche **Szenarien** durchgespielt:
  - Austausch der View-Komponente durch **GUI**
  - Auslagerung eines Akteurs (Rater/Codierer) ins **Netzwerk**
  - **Änderung** der Farben, Codelänge oder Formate

Architektur im Wandel

Initiale Überlegung

- **3-Schicht-Architektur** mit den Schichten **View**, **Logic**, **Data**
- View-Layer **validiert** und sendet User-Input an den Logic-Layer
- Logic-Layer **verarbeitet** Input und sendet Speicherstände in Data-Layer
- Data-Layer **speichert** Spielstände und stellt Modelle bereit

Spätere Erweiterung

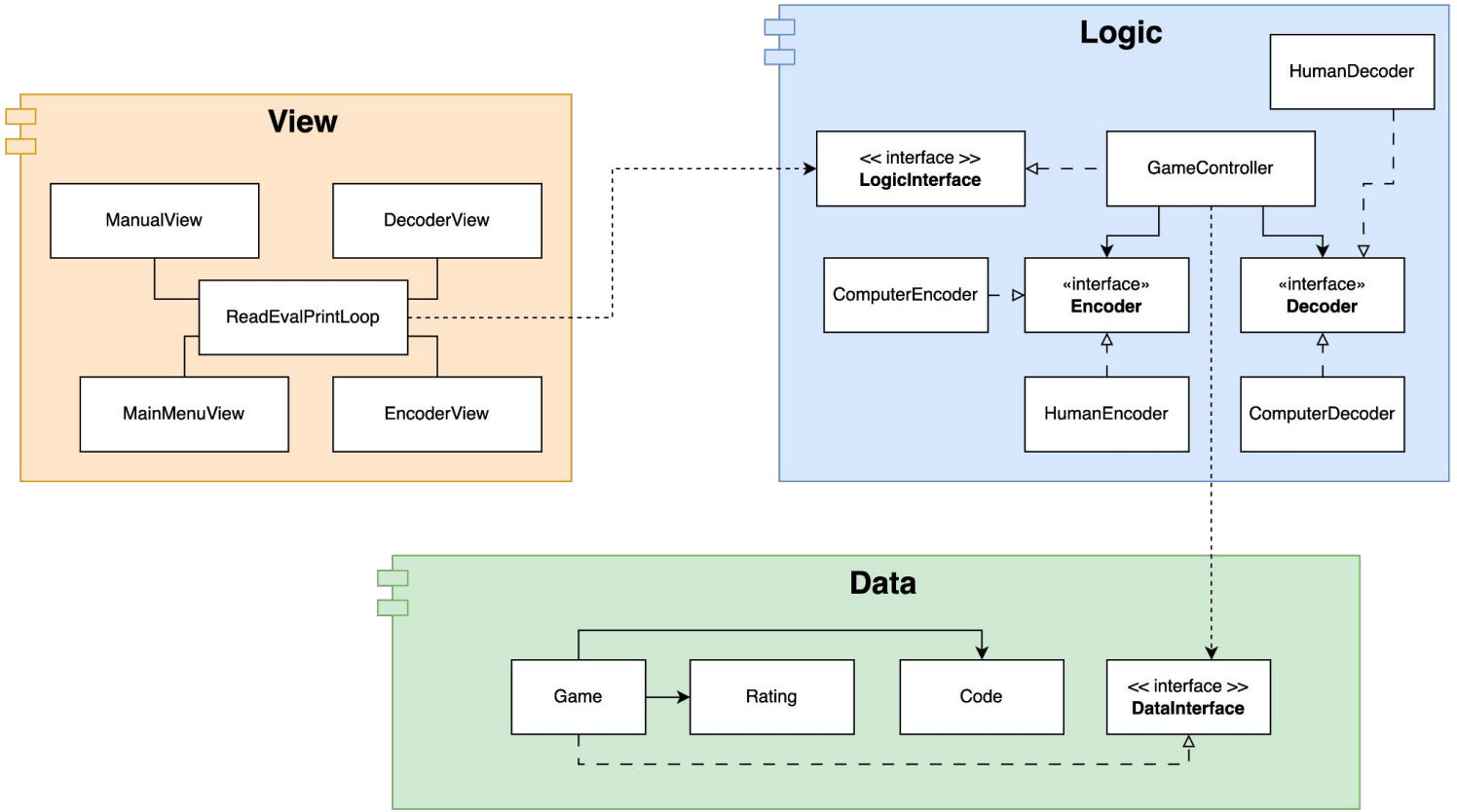
- Zu hohe **Kopplungen** zwischen Logic und Data-Layer
- **Erweiterung** und **Verlagerung** der Models in den Logic-Layer
- Verwandlung des Data-Layers in reines **Repository**

Ergebnis

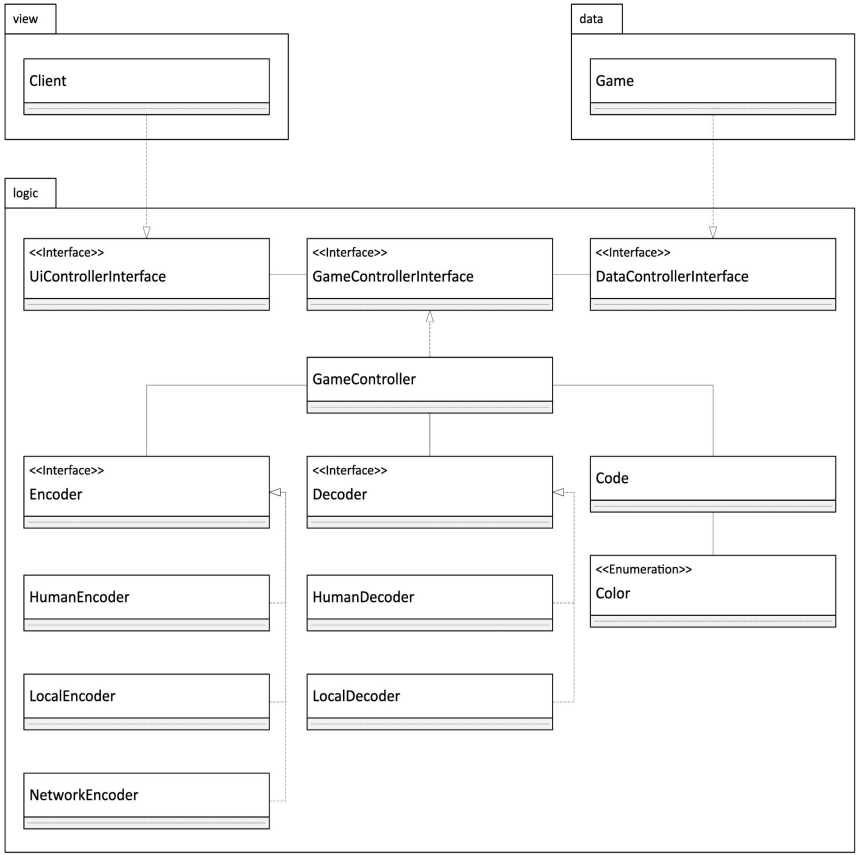
- **Mischform** aus Schichten- und Repository-Architektur

htw.

Grobentwurf: Initiale 3-Schicht-Architektur



Grobentwurf: Finale Architektur

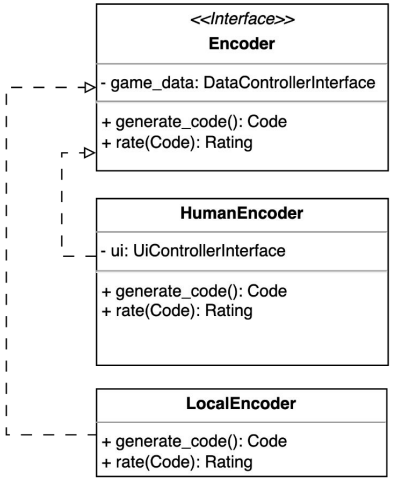


Feinarchitektur: Modulare Akteure

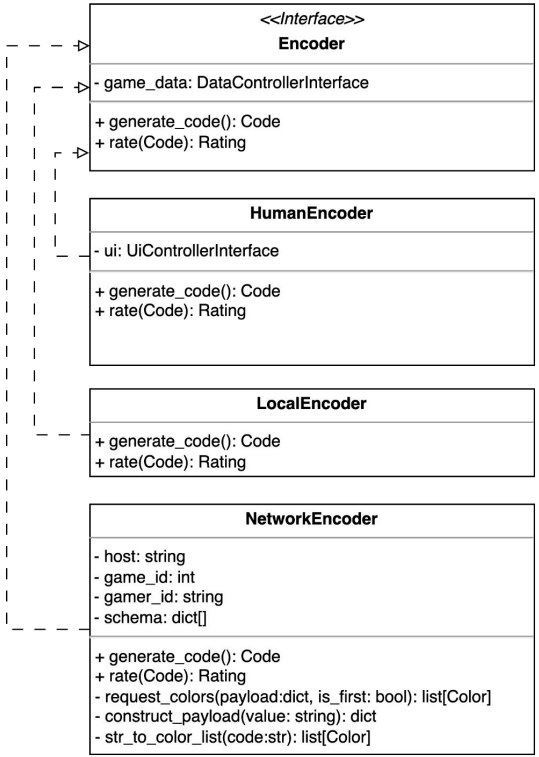
- Zwei zentrale, logische Akteure: **Rater und Codierer**
  - im Code: Decoder (Rater) und Encoder (Codierer)
- Zu Beginn bereits **polymorph**, da zwei mögliche Ausprägungen:
  - vom **Menschen** gespielt
  - durch **Computer** gesteuert
- Funktionen jedoch **nach außen hin identisch**
  - Encoder kann Codes erstellen und Versuche bewerten
  - Decoder kann Rateversuche abgeben
- **LocalEncoder** und **HumanEncoder** implementieren Interface **Encoder**
- **LocalDecoder** und **HumanDecoder** implementieren Interface **Decoder**
- Später erweitert durch den **NetworkEncoder**
  - beherbergt die Netzwerkfunktionalität
  - Agiert als Adapter zwischen lokaler Logik und Codierer-Server

Encoder-Feinentwurf

vor Netzwerk-Komponente



nach Netzwerk-Komponente



2. Nicht-funktionale Anforderungen

Hochschule für Technik
und Wirtschaft Berlin
University of Applied Sciences

Nicht-funktionale Anforderungen

- *“Als **Kunde** möchte ich innerhalb des Programms alle **Variablen**, **Kommentare** und **Dokumentationen** auf **Englisch** haben, um internationale Lesbarkeit und Erweiterbarkeit zu garantieren.”*
  - Durch die in **PyCharm** enthaltene **Rechtschreibprüfung** und während der **Code Reviews** wurde stets auf englische Sprache geachtet

- *“Als **Kunde** möchte ich das durchweg auf **idiomatisches Python** gem. der PEP-Richtlinien geachtet wird.”*
  - Unter Zuhilfenahme der **.editorconfig**-Datei und PyCharms **Formatter-Settings** wurden entspr. Conventions festgelegt

Nicht-funktionale Anforderungen

- *“Als **Kunde** möchte ich ein einzelnes Artefakt bekommen, das ich dann **per Konsole mittels Python 3** starten kann, um einfachen Zugang zum Spiel zu haben.”*
  - Es wurde ausschließlich **Python-Code** geschrieben und es sind lediglich **Python3** und dessen Bestandteile zur Ausführung nötig

- *“Als **Spieler** möchte ich die Software unter **Linux**, **MacOS** und **Windows** nutzen können, um die Bindung an ein Betriebssystem zu vermeiden.”*
  - Projekt kann auf allen Geräten mit gültiger Python3-Installation gestartet werden. Python 3 ist **für alle o.g. Betriebssysteme** frei verfügbar

- *“Als **Spieler** möchte ich, das **ungültige Angaben** erkannt werden und keinen Programmfehler zur Folge haben”*
  - View-Schicht **validiert** den User-Input und gibt **Hilfestellung**

3. Qualitätssicherung in der Praxis

Hochschule für Technik
und Wirtschaft Berlin
University of Applied Sciences

QS-Maßnahme 1: Git Workflow

- Genereller, standardisierter **Git-Workflow**
  a. Feature-Branch vom Main-Branch abzweigen
  b. Feature im Paar bearbeiten
  c. Code auf Remote-Repository pushen
  d. Merge Request öffnen und Reviewer zuweisen
  e. Nach MR-Approval in den Main-Branch mergen
- **Main-Branch-Protection**
  - Direktes Pushen auf den Main-Branch ist nicht möglich
- **Review Policy**
  - Merge erst nach 2 unabhängigen Approvals möglich

QS-Maßnahme 2: Partnerarbeit

- Code wurde grundsätzlich immer im **Pair-Programming** geschrieben
- Wo möglich wurde das **Ping-Pong-Programming** genutzt
  - Partner A programmiert die Unit-Tests
  - Partner B programmiert die entsprechende Implementierung
  - Iteratives Vorgehen
- **Reviews** wurden auch immer **paarweise** übernommen
  - Mitarbeiter A und B öffnen einen Merge Request
  - Mitarbeiter C und D übernehmen den Review
  - Doppeltes Vier-Augen-Prinzip
- **Rotation** der Partner
  - Vermeidung von Betriebsblindheit
  - Mehrere Perspektiven
- Wöchentliche Termine zwecks **Retrospektive** und **Review**
  - Jeden Sonntagabend via Video-Call

QS-Maßnahme 3: Unit-Tests & Pipelines

- Der Code wurde mittels **automatisierter Unit-Tests** getestet
  - Pro Komponente eine **Test-Suite**
  - Pro Test-Suite mehrere **eng gefasste Testfälle**
- **Vor jedem Commit** sicherstellen, dass die Tests durchlaufen
  - Lokale Ausführung
- Bau einer **CI-Pipeline** zur automatisierten Testdurchführung
  - Integration in das GitLab-Repository
  - **Problem**: dedizierter Runtime-Server benötigt
  - Deshalb leider nicht von Erfolg gekrönt

4. Reflexion & Optimierungspotenzial

Hochschule für Technik
und Wirtschaft Berlin
University of Applied Sciences

Was würden wir anders machen?

- Klare **Deadlines** setzen und diese auch einhalten
  - Häufige Last-Minute-Contributions
- Weniger **Interpretieren** und **Vermuten** – Mehr **Nachfragen**
  - Manche Architekturentscheidungen auf Annahmen basiert
- Den **Feinentwurf** während der **Implementierung** anpassen
  - Häufige Diskrepanzen zwischen Praxis und Theorie
  - Pläne können sich ändern und das ist auch okay so
- **Infrastruktur** und **Automatisierung** von Anfang an planen
  - Aufwand wächst mit Projektgröße
  - CI-Pipeline-Debakel verhindern
- **KISS** – **K**eep **I**t **S**imple **S**tupid
  - Komplexität heißt nicht gleich Qualität

5. Screenshots

Spielablauf als Rater

```
------------------------------------------------
Willkommen zu Super Super Hirn
------------------------------------------------
'help' zum Anzeigen der Spielanleitung
'exit' zum Beenden
------------------------------------------------
Das Spiel beginnt automatisch

Wählen Sie eine Rolle 'Codierer' oder 'Rater':
```

```
------------------------------------------------
Willkommen zu Super Super Hirn
------------------------------------------------
'help' zum Anzeigen der Spielanleitung
'exit' zum Beenden
------------------------------------------------
Das Spiel beginnt automatisch

Wählen Sie eine Rolle 'Codierer' oder 'Rater': rater
Lokal oder Netzwerk: lokal
Bitte Code Länge wählen (4 oder 5): 4
Anzahl der Farben wählen, 2 bis 8 möglich:
```

```
------------------------------------------------
Spielfeld
------------------------------------------------
Code: XXXX

[2, 2, 1, 3] | [8, 8, 8, 8]
[1, 2, 3, 2] | [8, 7, 7, 7]
[1, 2, 2, 3] | [8, 8, 7, 7]
[1, 2, 3, 3] | [8, 8, 7]
[1, 2, 1, 2] | [8, 8, 7]
[1, 2, 1, 3] | [8, 8, 8]
------------------------------------------------
Gewonnen [2, 2, 1, 3]
```

Spielablauf als Codierer

```
------------------------------------------------
Willkommen zu Super Super Hirn
------------------------------------------------
'help' zum Anzeigen der Spielanleitung
'exit' zum Beenden
------------------------------------------------
Das Spiel beginnt automatisch

Wählen Sie eine Rolle 'Codierer' oder 'Rater':
```

```
------------------------------------------------
Willkommen zu Super Super Hirn
------------------------------------------------
'help' zum Anzeigen der Spielanleitung
'exit' zum Beenden
------------------------------------------------
Das Spiel beginnt automatisch

Wählen Sie eine Rolle 'Codierer' oder 'Rater': codierer
Bitte Code Länge wählen (4 oder 5): 4
Anzahl der Farben wählen, 2 bis 8 möglich: 3
RED:1, GREEN:2, YELLOW:3
Bitte gebe einen Code ein: 1331
```

```
Spielfeld
------------------------------------------------
[1, 3, 3, 1] wird gesucht.

[1, 3, 3, 1] | [8, 8, 8, 8]
[3, 2, 3, 2] | [7, 8]
[1, 1, 2, 2] | [8, 7]
------------------------------------------------
Gewonnen [1, 3, 3, 1]
```

Abfangen von Fehlerfällen

```
--------------------------------------------------
Willkommen zu Super Super Hirn
--------------------------------------------------
'help' zum Anzeigen der Spielanleitung
'exit' zum Beenden
--------------------------------------------------
Das Spiel beginnt automatisch

Wählen Sie eine Rolle 'Codierer' oder 'Rater': zuschauer
Ungültige Eingabe. Wählen Sie eine Rolle 'Codierer' oder 'Rater':
Wählen Sie eine Rolle 'Codierer' oder 'Rater':
```

```
Bitte Code Länge wählen (4 oder 5): 6
Ungültige Eingabe. Bitte wählen Sie 4 oder 5.
Bitte Code Länge wählen (4 oder 5): 6
Ungültige Eingabe. Bitte wählen Sie 4 oder 5.
Bitte Code Länge wählen (4 oder 5):
Ungültige Eingabe. Bitte wählen sie '4' oder '5'.
Bitte Code Länge wählen (4 oder 5): 4
Anzahl der Farben wählen, 2 bis 8 möglich: 1
Ungültige Eingabe. Bitte wählen Sie eine Zahl zwischen 2 und 8.
Anzahl der Farben wählen, 2 bis 8 möglich: 9
Ungültige Eingabe. Bitte wählen Sie eine Zahl zwischen 2 und 8.
Anzahl der Farben wählen, 2 bis 8 möglich: 3.4
Ungültige Eingabe. Bitte wählen Sie eine Zahl zwischen 2 und 8.
Anzahl der Farben wählen, 2 bis 8 möglich: 4
```

```
--------------------------------------------------
Willkommen zu Super Super Hirn
--------------------------------------------------
'help' zum Anzeigen der Spielanleitung
'exit' zum Beenden
--------------------------------------------------
Das Spiel beginnt automatisch

Wählen Sie eine Rolle 'Codierer' oder 'Rater': zuschauer
Ungültige Eingabe. Wählen Sie eine Rolle 'Codierer' oder 'Rater':
Wählen Sie eine Rolle 'Codierer' oder 'Rater': rater
Lokal oder Netzwerk: netzwerk
IP-Adresse und Port im Format 'IP:Port' : 127001:3000
Ungültige Eingabe. Bitte geben Sie eine gültige Adresse im Format IP:Port ein.
IP-Adresse und Port im Format 'IP:Port' :
```

Vielen Dank.
Zeit für Ihre Fragen!

www.htw-berlin.de

Hochschule für Technik
und Wirtschaft Berlin

University of Applied Sciences

Hochschule für Technik
und Wirtschaft Berlin

University of Applied Sciences

www.htw-berlin.de