**1. McGee (2007):**

**1.1 In der BA**

Characteristics of patterns:

* Operational and precise
* Positive
* Flexible
* Debatable (the Pattern is clear enough to criticize)
* Testable
* End-user oriented

Patterns “express a relationship between particular design contexts, forces (psychological, social, or structural constraints), and desired (‘positive’ or good) features”

Sagt, es gibt wenig darüber, wie man neue Patterns macht, Innovation. Probiert innovativen Ansatz

**1.2 Im Original:**

Method for creating new patterns (Alexander):

* Notice a situation “where one feels good.”
* Identify the cause.
* Articulate it in a way so the feature’s presence in other structures can be binarily identified.
* Identify “the conflicting Forces it resolves.”
* Identify relevant contexts
* Empirical tests (of the reactions presence and absence of the feature causes)

Taught course on the design of computer games:  
Template:

* Name
* Forces (maximum of 2, sth people care about, “Do not include “solutions” in the descriptions of Forces. Rather, one force usually expresses a problem that happens if we “go too far” with the opposite of the other force.”)
  + Force 1: if a game does not have/allow [A], then players will experience problem [X].
  + the word “But,”
  + Force 2: if a game does have/allow [A], then players will experience problem [Y ].
* Feature (something we should make
  + the word “Therefore,”
  + a word that means making or building or doing
  + the specific rule to follow (feature) that will allow designers to resolve the conflicts (forces) described.)

“As context for this activity, students were advised that Patterns are initially weak hypotheses. We need to develop them to the point where they are strong enough to test – and then we need to start testing them.”

Evaluation checklist:

* Is the Pattern really present in the original game?
* Do you “believe” the Pattern – does it express something about the game that actually makes the game fun to play?
* Is it well-described:
* Forces
  + Is each Force a real Force (that people really care about)?
  + Is each Force relevant to the game?
  + Are the Forces in conflict?
    - Warning! One Force is not a “solution” to another Force. Rather, one Force usually expresses a problem that happens if we “go too far” with the opposite of the other Force.
* Feature
  + Does the Feature actually resolve the conflict in the Forces?
  + Is the Feature expressed as something we can do? (example: “therefore, do/make X”)
* Name
  + Does the Pattern name clearly express the Feature we should build?
* Do you find yourself nodding in agreement as you read the Pattern description?
* “If I had to use this Pattern to build something, would it help? Would it help enough?”
* Does the Pattern suggest interesting ways to improve the game?
* Is Pattern clear enough that we can separate games that have the Pattern from those that do not?
* Would games of the same type that do not have the Pattern be more fun to play if they did?

**2. Borchers (2001):**

Layout von Alexander et al. (1979):   
“A meaningful, concise name identifies the pattern, a ranking indicates the validity of the pattern, a picture gives a 'sensitizing' and easily understood example of the pattern applied, and the context explains which larger patterns it helps to implement. Next, a short problem statement summarises the competing 'forces', or design tradeoffs, and a more extensive problem description gives empirical background information and shows existing solutions. The subsequent solution is the central pattern component. It generalises the examples into a clear, but generic set of instructions that can be applied in varying situations. A diagram describes this solution and its constituents graphically, and references point the reader to smaller patterns that can be used to implement this pattern.” (S.361)

Semantics für sein formal syntactic Modell (Zitat S. 364):

* Each **pattern** of a language captures a recurring design problem, and suggests a proven solution to it. The language consists of a set of such patterns for a specific design domain, such as urban architecture.
* Each pattern has a **context** represented by edges pointing to it from higher-level patterns. They sketch the design situations in which it can be used. Similarly, its **references** show what lower-level patterns can be applied after it has been used. This relationship creates a *hierarchy* within the pattern language. It leads the designer from patterns addressing large-scale design issues, to patterns about small design details, and helps him locate related patterns quickly.
* The **name** of a pattern helps to refer to its central idea quickly, and build a vocabulary for communication within a team or design community. The **ranking** shows how universally valid the pattern author believes this pattern is. It helps readers to distinguish early pattern ideas from truly timeless patterns that have been confirmed on countless occasions.
* The opening **illustration** gives readers a quick idea of a typical example situation for the pattern, even if they are not professionals. Media choice depends on the domain of the language: architecture can be represented by photos of buildings and locations; HCI may prefer screen shots, video sequences of an interaction, audio recordings for a voice-controlled menu, etc.
* The **problem** states what the major issue is that the pattern addresses. The **forces** further elaborate the problem statement. They are aspects of the design that need to be optimised. They usually come in pairs contradicting each other.
* The **examples** section is the largest of each pattern. It shows existing situations in which the problem at hand can be (or has been) encountered, and how it has been solved in those situations.
* The **solution** generalises from the examples a proven way to balance the forces at hand optimally for the given design context. It is not simply prescriptive, but generic so that it can generate a solution when it is applied to concrete problem situations of the form specified by the context.
* The **diagram** supports the solution by summarising its main idea in a graphical way, omitting any unnecessary details. For experts, the diagram is quicker to grasp than the opening illustration. Media choice again depends on the domain: a graphical sketch for architecture, pseudo-code or UML diagram for software engineering, a storyboard sketch for HCI, a score fragment for music, etc.

“We will not describe the patterns in full detail; that would typically require **several pages per pattern**.” (S.368)

“The actual patterns are written in a more detailed textual form without explicit labels for 'Context', 'Problem' and so on: instead, they use implicit typographical structuring to clearly show the components of each pattern.” (S. 368)

*Beispiele bestehen aus Name, Context, Problem, Solution, (Image (not specified, optional)), Examples, References*

**3. Sharma et al. (2016):**

**3.1 In der BA**

(Sharma et al., 2016) mapped high level functions to low level functions and the latter to associated sensors. The paper also provides an overview of “the state-of-the-art sensors in terms of their technical specifications, possible limitations, standards, and platforms.” Furthermore the paper presents challenges associated with linking different kinds of sensors in a system, such as incompatibility with each other or the system architecture, data synchronization and amount of data.

**3.2 noch nicht**

Thematically, the work by (Sharma et al., 2016) can serve as a basis for the framework. They mapped low level functions such as gaze, voice, or hand gestures to associated sensors for use with Augmented Reality and provide an overview of common issues different sensors might encounter. The design synthesis approach utilized in the paper may inspire work on this framework. A list of transfer mechanisms, although intended for expertise transfer, could also prove useful. Finally, they provide a list of challenges in combining different sensors in one system; compatibility with other sensors (for example interference caused by multiple devices using infra-red light) is an example of a category that differentiates the sensor-supported game mechanism framework from previous work.

*Unterscheiden Sensoren: Smart glasses, Smartwatches, Point of view camera (add-on/alternative to Smart Glasses), Microphone, Body posture, Sensor-based systems (Myo), Eye tracking, EEG Brainwave sensors*

“Taking into consideration all the factors, **Microsoft Hololens** with features including: environment capture, gesture tracking, mixed reality capture, Wi-Fi 802.11ac, and fully untethered holographic computing, is the best candidate for the project. However, as Hololens is quite new (at the time of writing of this report), it has not been tested for compatibility with other sensors, as a source of interference for other sensors, and connectivity with different standards/devices. Furthermore, extensive testing is needed to see if Hololens can be used as a stand alone system for capturing all the data (both raw and processed) from different sensors.

”

“In the design of a system recombining the various different sensors identified above (all using different data rates and different standards for storage and communication), several notable challenges arise:

* + Compatibility and support of Unity development engine across different hardware sensors
  + Support of sensors across different operating systems and programming platforms
  + Compatibility of the different hardware drivers associated with the sensors.
  + Interference due to, e.g., noise generated by sensors.
  + Local and efficient storage of raw and processed data of the various sensors.
  + Synchronization of data owing to different data rates of the sensors (e.g., EEG, Augmented reality glasses, microphone).
  + Compatibility of the communication standards and protocols (for instance, Bluetooth, and WiFi) and their data transmission range.
  + The computational complexity and processing load needed for processing the data associated with different sensors.
  + Design of the WEKIT capturing system that integrates all the sensor hardware as a wearable system.” (pp. 37-38)

(Nicht alle hier sinnvoll (Vor allem Tasks irrelevant, low-level + sensor geht))  
-> Andere WEKIT Quelle evtl besser direkt anwendbar

|  |  |  |  |
| --- | --- | --- | --- |
| **Relevant Tasks** | **Low-level functions** | **Sensors** | **Key Products** |
| Tele-assistance | View and capture the activity of another person from their perspective: transmit video & audio. | Smart/ augmented reality glasses | Moverio BT-200/2000, Microsoft Hololens,  Sony SmartEyeglass,  Google Glass, Meta 2,  Vuzix M-100, Optinvent Ora-1,  ODG R7. |
| Tele-assistance, realtime feedback, | Capture from the perspective of the user. | Point of view camera | GoPro Hero, Panasonic A500 Camera, Smart Glasses    Custom integrated camera |
| Think aloud, Remote symmetrical tele-assistance | Capture voice of the user. | Microphone | Cochlea Wireless Mini Microphone, built-in microphone of Camera/Smart Glasses, Wireless Microphones (e.g. from AKG) |
| Remote symmetrical tele-assistance | Capture and model animation of hand movement or gestures | Optical tracking using depth scanner, Smart armband sensing muscle movement | Myo Gesture control armband, Leap Motion controller |
| Contextualisation, in situ realtime feedback, virtual post its | Object tracking in environment | Smart glasses, Tablet Computer or Mobile Phone (all + AR tracking toolkits, e.g. Vuforia, Alvar, ARToolKit) |  |
| Contextualisation, in situ realtime feedback, virtual post its | Location tracking in environment | Outdoors: GPS  Indoors: wifi triangulation, beacons, optical location tracking | Find the direction of the object: Beacons such as Estimote or Tile    Locate object: computer vision + AR tracking  Find the location on a map: GPS |
| Virtual/ tangible manipulation | Hand movement tracker, accelerometer, gyroscope | Depth camera, smart armband | Myo Gesture control armband, Leap Motion controller, Smart Glasses |
| Haptic hints | Vibrations on arm or fingers | Vibrotactile bracelets | (MYO), magic ring |
| Virtual Post Its | Place and see virtual post its | Smart Glasses / Tablet computer |  |
| Mobile Control | Control dials / other UI elements | Computer vision | Hand controller API for Unity (e.g. Augmenta) |
| In Situ Real Time feedback | Provide step by step instruction | Computer vision, activity detection | Bespoke software solution in AR software |
| Case Identification | Link with existing cases, link with error knowledge | CBR reasoning component | No sensor required |
| Directed focus | Direct focus of technician | Gaze direction / object recognition  EEG (attention/focus/mental effort) | Smart Glasses (or gyroscope only)    MyndPlay MyndBand  Interaxon Muse EEG  Neurosky Mindwave  Emotiv EEG |
| Directed focus | Reduce distraction | Eye tracking,  EEG for attention monitoring | Attention Protocols in EEG are ideal for this, any of the above EEG    SMI eye gaze tracker, Tobii eye tracker, Eyetribe eye tracker, Pupil labs eye tracker |
| Self-awareness of physical state | Fatigue level, vigilance level | EEG (e.g. p300 response)    Papers on EMG with GSR    GSR has best references and more published papers for fatigue | MyndPlay MyndBand,  Neurosky Mindwave  Emotiv    Readiband system as used by BBMV  EMG through EEG |
| Self-awareness of physical state | Time on task,  time of day (morning shift, evening shift) | Can be done by AR vision system |  |
| Self-awareness of physical state | Capture body posture: ergonomics (e.g. lean back, forward)    Capture gestures and movements (hand positions, finger positions) | Camera-based systems for non-wearables, gyroscope, accelerometer, magnetometer for wearables | Kinect, RealSense, LeapMotion, Myo, check alternative body-worn system on spine (Lumo), |
| Self-awareness of physical state | Biodata (like steps, sleep, heart rate, GSR) | Smart wristband, smart watch  Smart shirt | Fitbit, apple watch, other health devices  Multiple sensors available |
| Contextualisation | Recognise environment | Smart glasses or other AR camera    Meta-data model for contextual data | See Smart Glasses |
| Object enrichment | Recognise objects,  augment them | Smart Glasses, Tablet computer |  |
| Object enrichment | State proofing | Smart objects | Fit objects with Arduino logic, Raspberry Pi, microbit |
| Zoom | Zoom in and get details | Smart glasses / tablet with high resolution camera    Light field camera    360 cameras | See above |
| Slow motion | Allow replay at slower speed | High frame rate camera (warning: high frame rate often comes at price of resolution with smart glasses; and vice versa) |  |

**4. Wetzel:**

**5. Antonaci:**

**6. Schmitz:**

**7. Andere?:**

Anderes außer Patterns: Definition von AR, Evtl. Anwendungen als Grundlagen für Patterns, …

Roland:

Vorgehensweise für das Framework: ich würde versuchen, basierend auf Deinem Literaturergebnis Bestandteile des Frameworks auszuwählen und zusammenzustellen, die speziell sind für AR oder spezifische Probleme von AR lösen. Dabei kannst Du bewusst eine Einschränkung vornehmen auf Hololens (und vergleichbare Geräte) als Plattform. Das führt Dich z.B. zur Diskussion von folgenden Aspekten: - Game design (erstellen von Spielen für unbekannte Räume, Spiele mit/ohne Bezug zum Raum) - Interaktion (Steuerung, Navigation, Auswahl) - Visualisierung (HUD, Navigationshilfen, virtuelle Objekte und ihre Platzierung, NPCs) - optional: Multiplayer-Fragen? (z.B. sharing von Hologrammen? Was wäre mit remote-multiplayer?) Daraus lassen sich dann Vorschläge/Konzepte ableiten, von denen ausgewählte in Deinem Prototypen landen