**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

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)

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:**

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):**

**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