Bachelorarbeit am Institut für Informatik der Freien Universität Berlin
Human-Centered Computing (HCC)

An adversarial interface design pattern to support ideation

Gerold Schneider

Betreuer: Maximilian Mackeprang
Erstgutachterin: Prof. Dr. Claudia Müller-Birn
Zweitgutachter: Prof. Dr. Lutz Prechelt

Berlin, July 30, 2019

Eidesstattliche Erklärung

Ich versichere hiermit an Eides Statt, dass diese Arbeit von niemand anderem als meiner Person verfasst worden ist. Alle verwendeten Hilfsmittel wie Berichte, Bücher, Internetseiten oder ähnliches sind im Literaturverzeichnis angegeben, Zitate aus fremden Arbeiten sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungskommission vorgelegt und auch nicht veröffentlicht.

Berlin, July 30, 2019

Gerold Schneider

Abstract

This project has identified a software user interface design pattern that captures the essential nature of problem oriented ideation processes and possibly improves their outcome and efficiency by providing mechanisms to overcome common problems encountered in such processes. In particular creative block, the blank page, loss of focus and methodical constraints. As far as the yet limited evaluation of the pattern through a prototype implementation and a hand full of expert inquiries allows valid inference, its utility is clear and demonstrable. Of particular note seems to be the observation that the pattern does not formulate or enforce a particular method or paradigm but instead provides just enough structure for the user to feel secure while being flexible enough to become an extension of any users personal method. It is therefore well received by users that generally reject method-driven creativity. In contrast to existing tools (offline and online) this pattern seems to resolve all critical issues that constrain productive use (constrained space, problem-solution mismatch, cost of complexity, overhead), leaving only issues that could improve its standalone effectiveness and complement value through integration into existing workflows. The pattern appears to be fairly agnostic in terms of ideation subject matter, even as far as being adaptable to tasks ancillary to ideation like note-taking, collecting materials, sharing and browsing materials, documenting, sense-making and clustering.

The pattern reaches its limits where users are unfamiliar with concepts of creativity or have little experience in solving problems creatively. The pattern cannot overcome inherent lack of creativity or disinterest in solving a problem. As a pattern aimed at design, its complementary value when integrated in production tools is unclear.

Zusammenfassung

Dieses Projekt identifiziert ein Gestaltungsmuster für Software-Benutzerschnittstellen welches den essenziellen Charakter von Problemgerichteten Ideenfindungsprozessen einfängt und ihre Ergebnisse verbessern kann indem es Mechanismen zur Abmilderung gängiger Probleme solcher Prozesse anbietet. Diese sind insbesondere Blockaden des kreativen Denkens, Schwierigkeiten einen Anfang zu finden, Verlust des Fokus und die gegebenenfalls engen Beschränkungen einer bestimmten Methode. Insofern die noch stark beschränkte Auswertung einer prototypischen Implementierung des Gestaltungsmusters und eine Handvoll Expertenmeinungen gültige Schlussfolgerungen zulässt ist, hat es sich als durchaus nützlich gezeigt. Insbesondere die Beobachtung scheint bedeutsam, dass das Gestaltungsmuster keine Methode, bestimmtes Vorgehen oder Paradigma fordert, sondern Anwendern grade genug Struktur bereitstellt, dass sie sich sicher fühlen und mit ihm ihre eigene Methode ergänzen. Es wurde deshalb auch von Anwendern gut auf genommen die sonst methodische Ansätze der Kreativitätsförderung ablehnen. Im Gegensatz zu existierenden Anwendungen scheint das Gestaltungsmuster alle kritischen Probleme zu lösen welche einer produktiven Nutzung entgegenstehen (Beschränkter Raum, Problem-Lösungs-Diskrepanz, Komplexitätskosten, Verwaltungskosten). Es verbleiben lediglich Belange welche seine selbständige Effektivität und seinen Wert bei Integration in existierende Arbeitsabläufe verbessern. Das Muster scheint durchaus unabhängig vom dem Gegenstand einer Ideenfindung nützlich zu sein und eine gewisse Flexibilität zu besitzen für Nebenzwecke einer Ideenfindung, wie z.B. Notizen, Materialsammeln, Dokumentation, Sensemaking und Clustering.

Das Muster erreicht die Grenzen seiner Nützlichkeit wenn Anwender kein Verständnis für Konzepte der Kreativität oder wenig Erfahrung mit kreativer Problemlösung besitzen. Das Muster ist nicht in der Lage ein inhärentes Fehlen von Kreativität oder Desinteresse an einer Problemlösung zu überwinden. Als ein Muster dessen Nutzen sich vordergründig als Entwurfs- und Gestaltungswerkzeug entfaltet, ist der Nutzen z.B. bei Integration in Produktiv-Werkzeuge unklar.

Contents

List of Figures

List of Tables

1 Preface

Supplementary materials

The sources, builds and other materials created over the course of this project are hosted in a Bitbucket repository and publicly accessible. In particular standalone builds of the prototype implementation for macOS and Windows are included. The repository can be accessed at the following web address (URL):

`https://bitbucket.org/gerolds/periscope05/`

2 Introduction

2.1 The role attributed to computational artifacts

A common reduction of computing and its primary utility is: *Computers just make things faster.* While not entirely accurate this statement reflects the predominant aspect that is leveraged in computational artifacts and, in one way or another, subject of most research in theoretical computer science[1]. From the side-effects such artifacts have, their normative quality, it can be inferred that they also have a property, like all things that partake in our everyday activity, to influence our way of thinking. Yet this aspect, while frequently subject of research in terms of illustrating the moral responsibility that exists in the creation technology which drives contemporary and future society [Ramo6], is rarely explored as the subject quality of an artifact. We intuitively think of objects that invite us to think as educational gizmos, tedious, impractical and academic. Thinking, it seems, is not welcome in a routine world. Everything must be fast and intuitive and automatic. Best practices, general models, reactive behavior and all kinds of lost energy. This is not a bad thing in principle, but it starts to become one when instead of place and time, comfort becomes a tyrant in a technologized world. When thinking can be delegated to industrious experts and systems. When we expect nothing more from our tools than to give us quick answers, tools will be designed to do just that. This doesn't mean that the answers are bad or wrong. But it means that the questions that are asked are limited in scope and bounded by the expectations that we have of our tools. A great tool to answer question *A* invites many to ask that kind of question in favor of question *B* that is not so easily answered[2]. Over many instances this creates a trajectory towards ever less original[3] thought as integral part of our activities. **The question then is, how do we design artifacts that integrate a rewarding experience with properties of original thought in routine interactions;** such that they become an expected property of our tools.

As it stands now, computers commonly don't offer much choice in that regard to those who'd actually like to engage with differently. They generate reports, execute commands, facilitate communication but they rarely give us ideas, challenge our assumptions or engage with us in some sort of legitimate dialogue. Why

1 Education in computer science is framed largely by the notion of efficient calculation subservient to some hypothetical utility and pays little regard to questions of purpose other than abstract trade-offs against *"as fast as possible"*.

2 Cognitive bias and the law of the instrument: Answers, especially easy ones, have a rewarding property and positive affect [Kah11].

3 Original thought involves reflection, creativity, diversification, heterogeneity, skepticism, experiment, rationality, in-, de- and abduction, etc.

is that? Can't we design interactions that facilitate insight and understanding of things, far beyond what unaided thinking provides? Supposedly we have to make computers intelligent first. But is that actually so? When we share our ideas with others, is the intelligence of our opposite integral to us improving our thinking? Or is it the challenge in expressing our thought, in constraining it to various languages, in framing it differently, repeatedly, to make ourselves understood, what actually improves it? What role do we attribute to the object that is allowed to challenges us? Can we synthesize this challenge and make the computer an adversary we engage with [Lau93], that brings out the best in us? Can we build general metaphors and patterns around the idea of engaging in discourse, in exploration, in improving our understanding of a complex world by experimenting with it? Can we dance with a machine?[4]

2.2 Concrete goals

To make one small step towards a different mode of engagement with computers, the specific goal of this thesis project is the creation of one instantiated example of a HCI pattern that explicitly challenges assumptions, requires decisions and fosters explorative thinking with the purpose of improving understanding and creativity. In a sense, using informatics not to calculate answers but to pose interesting questions.

The prototype instantiation of this HCI pattern, is a viable desktop application that can be applied in any problem situation involving fragmented information, a vague goal and emergent structure. I.e. research, decision making, project management, idea review, studying, etc. Or practically any situation where the intended outcome is *better ideas* and especially conducting deliberate innovation of manmade things. For the sake of gleaning valid insights from the prototype on that end and to reflect on the efficacy of action research (chapter 3.4) for this purpose, the prototype is constructed in particular to be tested qualitatively by creativity experts at Fraunhofer Center for Responsible Research and Innovation (CeRRI). The prototype is codenamed Periscope 0.5.

This project also specifies the Stacktree pattern that facilitates this innovation and explorative thinking as exemplified by the Periscope 0.5 prototype.

A complete package for evaluating the Stacktree pattern through its Periscope 0.5 prototype in additional contexts is included in appendix E for the interested reader to try themselves in order to validate, differentiate or contradict the findings included here.

4 *Dance* is the expression of meaning through limited motion, interacting with a constrained space and constructively using the tension this creates to form drama. Used here as an analogy.

2.3 Approach

In reference to [Lau93] and [DiS15] this project supposes that, if information systems design is approached from a perspective of making these systems constructive adversaries, that user- and ownership-experiences conductive to creativity, original thought and innovation can be created. In designing a prototype software for this purpose, the concrete strategy used in this project to that end, is to create a set of virtual entities that exhibit clearly defined and easy to understand behaviors that would allow a user to utilize them in a range of processes, methods and environments that are not possible with physical artifacts (or simulations of them). The goal is to establish establish an environment where computer and user together, create a system that improves each others capabilities. It is not the goal of this project to turn a computer into a slave that does all the work for the user, but to give the user access to their inherent abilities and (new) ways of realizing their intentions, particularly in the context of situations asking for creativity, innovation or open ended deliberation. This involves multiple dimensions of design, in particular one that is concerned with the role attributed to the artifact in a deployment context.

If an artifact is supposed to be able to assume such a role on its own merit[5], this requires invention of component entities that respond predictably to each other and their environment, giving them an *inherently virtual physicality*, weight, resistance and character, which make them useful, constituent entities of that environment. To understand the systemic effects of their interaction, and by extension the quality they project onto the artifact they compose, they have to be grounded in a model of the environment they act on. Ideally they can be abstracted into a pattern to be used for adapting them to various situations without loosing their effectiveness.

2.4 Method

To arrive at an artifact such as the one described above, in this project, a theoretical model of the the general problem situation (explorative thinking, ideation for innovation, sense-making of complex systems) is combined with empirical observation of an ideation workshop, translated into a software design, instantiated in a prototype and evaluated by a group of ideation experts. The prototype is an experiment which aims to discern the qualities of the proposed model, the pattern underlying and their its efficacy in designing systems that promote innovation and creativity in routine interactions.

To evaluate the effect of the software in the context of professional work, a high function prototype is needed; Especially to prevent negative effects from missing basic functionality, that could potentially influence the overall assessment.

5 as opposed to being commissioned for that role by higher authority

Methodologically this project is framed by the idea of Action Research where a designed artifact is introduced into a system and the system's response to the artifact is examined. As method for this project this provides opportunity for design and instantiation of an information systems artifact and results in knowledge derived from observations of its effect [Bas99] [SHP+11].

The details of this method are described in chapter 3.4. Section 3.5 summarizes the explorative research conducted to foster understanding of the problem situation. Chapter 3 summarizes current understanding of the problem situation in related work. Section 3.6 describes the model that underlies the decisions of the subsequent prototype software design.

2.5 Process

The following chapters describe in detail the steps that were taken in the attempt to reach the goals described above. Figure 2.1 illustrates their sequence and delineation. All boxes in this diagram are covered by this text, the appendices and additional online resources (in particular the Periscope 0.5 prototype). The following is a brief overview of the steps taken.

Background A deliberate action starts on assumptions about a given problem situation. To make decisions that lead to actions traceable, the project starts with a description of those assumptions, a choice of method and exploratory research to ground the initial assumptions. Real world constraints are identified through the observation of a typical instance of the problem situation at CeRRI. Covered in chapter 3.

Constructs & Modeling Based on the insights acquired so far, a model is constructed that explains the dynamics of the problem situation (ideation for innovation). Covered in section 3.6.

Prototype Based on the model and explorative research the Stacktree pattern is defined and an implementation design is developed. Requirements and design goals for a minimally viable prototype implementation are specified. The implementation architecture is detailed from multiple perspectives and an evaluation protocol is developed [Sto10] [Bas99] [SHP+11]. Covered in chapter 4. A detailed look at technology selection and implementation details of the design's core features is included in appendix B.

Aggregation The evaluation data is reviewed qualitatively to identify flaws in the model, pattern and application design. The findings are summarized and opportunities for improvement are derived. Covered in chapter 5 and chapter 6.

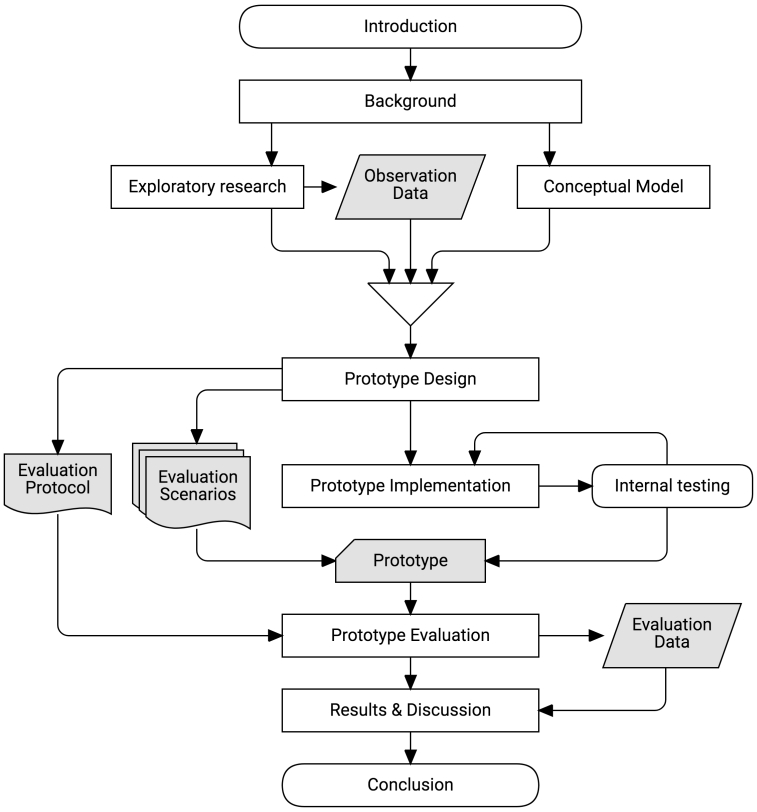

Figure 2.1: *The structure of this project, produced artifacts and topics covered in this text.*

2.6 A note on language and design of this text

This project was developed in bilingual setting where the planning, implementation, documentation and analysis was handled in English while experiments, observations and interviews were conducted in the native German or preferred working language of participants to minimize language barriers. To improve the readability of this text, all materials used in the main text have been translated into English where their understanding is necessary to follow the text.

To make this text accessible to readers from all disciplines the use of technical terms is in most cases paired with footnotes expanding on the mentioned concepts and clarifying meaning. Please note however that this text also draws on works from multiple disciplines and often concepts are mentioned and summarized only to sensitize the reader on their general idea as it is relevant to this project's core questions, without going into the detailed context in their originating discipline[6]. This may lead to some terms being used too loosely for a particular

6 Some overlap may occur where terms are shared between disciplines. For example the term "design" refers to various concepts depending on context and discipline. Additionally there exist individual opinions on what exactly its scope is. In this text then

expert's taste. It is my hope however, that the overall trajectory of the arguments surrounding such occurrences still clarifies their meaning sufficiently where footnotes have been omitted or forgotten.

"design" is never used as a technical term relating particularly to computer science or software engineering and always in its general, non specialized interpretation of *"purpose or planning that exists behind an action, fact, or object"* [ode10].

3 Background

3.1 Related work

Generally software tools[1] can be sorted into two fairly distinct categories: Production tools that are meant to create or manage structured data for publication[2]; And design tools in which the interaction with the tool is in itself the result. Production tools are more prevalent than design tools, however not for lack of design problems. And interestingly enough, production tools are routinely used as design tools. Textfiles are used for note keeping, logos are designed with layout software, spreadsheets are used for project planning, Computer Aided Design (CAD) tools are used for design exploration. And so far this seems pretty natural. After all they all have affordances that minimally support these hacks. What makes them effective as design tools however isn't just the provision of affordances. They also pose certain questions which must be answered. This forces reflection on the subject that the user wants to communicate. And it requires them to translate their ideas into specific decisions that, as a whole, capture what they meant to express. These decisions, the intuition and rationals behind them are however in themselves a complex network of experiences, concepts and information that is not usually captured, expressed or tracked by production tools.

For example: A spreadsheet offers boxes that can represent time-slots in a project, rows may represent resources, together looking much like the boxes on a paper calendar. However the dependencies between these boxes are only captured in the users's mind, if they want to iterate their planning, they have to remember all contingencies, the interesting ones and the menial ones. Ultimately this leads to project planning that is simplistic and maybe inefficient, just because spreadsheets were used for planning. Since this is a problem with a straight forward solution, sophisticated methods[3] and software[4] exist that support the scheduling of projects. These allow publication of simple views that communicate a schedule design to an audience but are actually a sandbox in which the project manager explores their options whenever conflicts arise.

There are various other attempts to create such tools. These are often languages, sandboxes and simulations that allow exploration of rules, flows and keeping

1 A tool is understood to be a thing that produces a valuable result through transformation of some medium.
2 Publication is here as an analogy describing any process where the result of work is passed on to others outside the original work environment.
3 I.e. Gantt charts
4 I.e. Microsoft Project

track of ideas. These are often very personal and niche tools tailored to specific problems. And often they are formal models and methods that are inaccessible to non-experts. In research such methods are used, provided the research team has resources to build their own environments that facilitate these tools. Examples include Petri-nets for exploring resource flows, UML for reflection on system architecture, TLA+[5] to explore behaviors of concurrent systems. Even programming itself can serve as an exploratory tool when supported by a suitable framework like MATLAB[6], RStudio[7], scientific mode in PyCharm[8] and in itself the Lisp family of languages that can, by virtue of their simplicity, be fully grasped and used to construct ones own novel ideas in code with minimal concern for how they are actually computed. Next to these powerful, but arcane instances, exist also a few accessible ones, dealing with rather more intuitive problems and incomplete information. These are less sandboxes as rather hacks to overcome the limitations of the human mind like limited rote and working memory, fatigue from slow thinking [Kah11], bias, limited rationality, motivation and so on. They include methods like brainstorming[9], card sorting[10], whiteboarding[11], mind mapping[12], clustering[13], role play[14], commonplacing[15] and the method of loci[16] among many others. A subset of these has been adapted and improved by software implementations that overcome some of the supposed problems these methods have in their physical form. Most of them however lack similarly overriding benefits that production tools have introduced to their underlying activity over low-tech methods. It appears that tools supporting design, explorative thinking and creativity remain elusive.

5 TLA+ is a high-level language for modeling programs and systems — especially concurrent and distributed ones. *Quoted from* `http://lamport.azurewebsites.net/tla/tla.html`, *retrieved (July 30, 2019)*

6 MATLAB is a numerical computing environment. MATLAB is also a domain specific language for numerical computation.

7 RStudio is an integrated development environment for the R language. R is a domain specific language for statistical computing and graphics.

8 PyCharm is an integrated development environment for the Python language. Python is a general purpose scripting language.

9 Brainstorming: Group creativity method to minimize inhibition, foster heterogeneity and idea transfer.

10 Card sorting: Research method to discover dominant category-associations of items in a group.

11 Whiteboarding: Method to improve idea and knowledge transfer by placing information on a shared canvas.

12 Mind-mapping: Multi purpose technique for exploring a subject by decomposing it into a hierarchy of ancillary concepts

13 Clustering: A method to reduce complexity by associating items with categories

14 Role play: Multi purpose method to explore interaction dynamics from alternate points of view

15 Commonplacing: The practice of compiling knowledge one acquires into books for later review based on a personal method.

16 Method of loci: Association of information with spatial memory of a familiar place.

3.2 The map metaphor and prior work

Prior to the start of this project I had been experimenting with various prototypes and problem situations that were designed to simplify sense-making over medium sized data sets. In particular information tokens that could take the form of index cards. All of these prototypes supposed that a map metaphor is conductive to sense-making as it is intuitively understood by many and provides an immediate perception of progress, creating positive affect that improves the retention of a tool all other things being equal.

As a starting point for this project, the most powerful features of a map metaphor, and by extension any spatial arrangement of objects, were identified as (1) providing orientation, (2) meaningful proximity relations and (3) the nearly invariant position of items. All these aspects are capable of critically effecting the utility of a map. Lack of orientation features makes a map tedious to use; meaningless proximity relations breaks the metaphor; And maps that change progressively loose their reference value, destroying orientation and the meaning of proximity. Additionally certain difficulties have been identified in the relation of the map metaphor to its supposed use case. Particularly visual complexity, sub-optimal accessibility of detail information, misunderstanding of visual cues and the utility of the design in the first place have been identified as problem areas. Ultimately the most significant insight of this whole process was the realization that low-fidelity prototyping and experiments with participants lacking domain expertise, are ineffective for coming up with an application design pattern that supports an activity which builds on such complex dynamic effects as sense-making does. The conclusion of these experiments was marked by the publication of [MSS+18], which briefly describes a design that is not sufficiently effective[17]. Despite this superficial failure certain new ideas suggested themselves about mapmaking as a mechanism for ideation, that are explored in this project.

3.3 Approach to application design

Application designs derived from a purely analytical view of requirements engineering and its tendency to decompose a system immediately into discrete units, run the danger of missing the actual problem [RM15] behind overly specific structures. There is no guarantee that the actual problem is ever specified clearly and solved effectively: An agglomeration of features does not automatically evolve into a coherent whole. This project acknowledges this problem and approaches the software design from a deliberately ambiguous "central idea" perspective: It is assumed that the designer of a system has sufficient control over variables determining the success of their hypothetical design, that they can solve the problem

17 Evaluation of prototypes up to this point suggested that application feel, the timed behavior of entities and actual content turned are essential for users to be able to grasp the design and evaluate it in a meaningful way.

by throughly understanding it and constructing a *comprehensive solution* that *systemically resolves all issues preventing actual utility*. If they do not have that level of control, their attempt can only succeed by accident.

Also, this project makes no concessions to any specific, isolated use case but aims, from the start, at devising a solution applicable in as general a case as possible. Generalizing concepts and observations removes distractions and makes the design more robust and flexible. Also this project acknowledges that the subject under study here is a design pattern and not the usability of any one feature. Therefore the minimal viable product at the end of this project must be one that fully captures the pattern, allowing for real evaluation of its utility. This probably requires something of a "vision" and can't be fully planned for or described outside building the thing itself.

3.4 Research method

3.4.1 Prior work

Before the start of this project, various approaches have been explored by the research group for Human-Centered Computing at FU-Berlin to understand and support ideation processes through information systems. In particular on the question of how to make a set of information fragments more accessible to a reviewer such that the outcome of ideation supported by these fragments is improved. Various avenues have been explored through means of rapid prototyping (paper, click-dummy, ...) that resulted in a concept design [MSS+18] for such interaction that has ultimately illustrated the limitations of rapid prototypes for exploring this particular instance of a complex problem. The key takeaways from this process were the following:

- Software that aims to support experts cannot be evaluated by non-experts
- A tool that purposefully transforms or leverages content cannot be evaluated in separation from actual content
- Tasks used in evaluation must be perceived as legitimate or real by the participant
- If application feel is critical for outcome, it cannot be simulated, abstracted or supposed
- Features cannot be tested in isolation if the evaluation aims at determining the utility of a given HCI pattern that involves multiple components.

In effect this means that an application that supports ideation requires a prototype of fairly high fidelity and feature completeness, with close to real content, legitimate test scenarios and and a number of domain experts for evaluating it. Based on this experience the method of this project instantiates the abstract user centered design (UCD) research method of context identification, requirements specification, iterative design & development and evaluation into a concrete form.

Also this and other prior experience together with an examination of a wide range of available tools (discussed in detail in 6) made it evident that computer based tools and methods in the context of ideation easily develop properties that negatively influence outcome quality. Following is a list of guide rails that are employed in this project to deal with them in the attempt to build something that works. While not individually attributable to specific related work they are largely informed by [Ben09], [Nor13], [Tuf01], [SZ03] and said experiences.

- Overcome the bad-content problem. Offer absurd, challenging and artful influences, provide an environment that feels safe and welcoming to an explorer.
- Overcome diminished creativity. Offer content that immediately skips over the most obvious avenues of thought, work hard on creating unexpected connections.
- Make the software useable and useful in practice. Make it a joy to use or at least make it get out of the way.
- Avoid UI tropes that require abstractions unnatural to the process of ideation. Use a custom GUI.
- Avoid the inconsequential vehicle of delivery problem. Create affordances that invite a process that is useful and inherently digital.
- Build a viable application that can be tested on actual problem situations.
- Maintain a scannable structure [Ben09].
- Minimize visual complexity.
- Provide efficient and precise controls.
- Make all relevant information directly accessible and editable.
- Abstract the visual appearance of everything as much as possible until it only signifies the affordances that are actually there.
- Make the application visually appealing.

Clearly however these aren't yet sufficient to make informed design decisions.

3.4.2 Action research

The framing of the actual method employed in this project is found in action research, a method used in the study of organizations, social behavior, pharmacology, medicine and others. It involves components of theory, taking action based on that theory and reflecting on the observed effects. More particularly it is described by [Bas99] and [SHP+11] in an application to information systems. In summary: An artifact is introduced into a system[18] and the system's response to the artifact is examined.

As method for this thesis project it provides opportunity for design and instantiation of an information systems artifact and results in knowledge derived from observations of its effect on a situated system. By designing the artifact tied to a

18 In this project: ideation processes

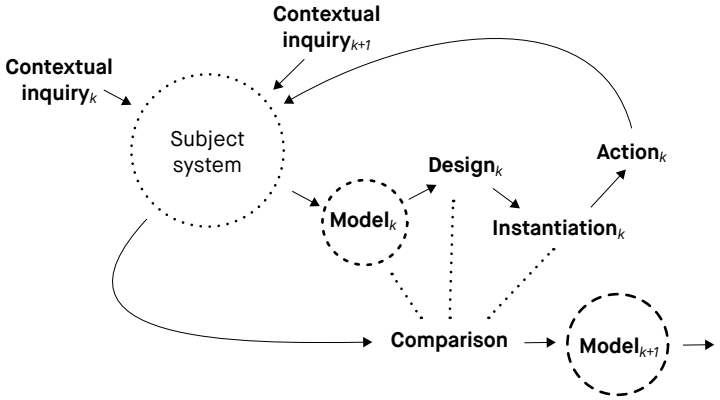

Figure 3.1: *Information systems action research loop*

model derived from observation of the subsequently to be disturbed system, design decisions can be traced in their effect[19]. Following is an outline of action research and the corresponding chapters in this text.

1. Discover the subject system (section 3.5).
2. Model the system. Combine theory and empirical observation. Use theory to know what to look for, and empiricism to figure out what the important elements are (section 3.6).
3. Design an artifact (chapter 4).
4. Build the artifact (appendix B).
5. Introduce the artifact into the system and observe (section 4.6).
6. Discern what effect the artifact had on the system (chapter 5).
7. Learn how the artifact can be improved (chapter 6).

19 Action research also provides a framing compatible to a qualitative method like contextual inquiry that is used as the primary approach to discovery and evaluation in the course of this project. I.e processes, interactions, problems and opportunities are observed and discussed cooperatively with participants that are actual stakeholders in the problem space yielding phenomenological insights into influences acting on the problem space as opposed to structural and factual influences that may or may not be perceived. In the context of ideas, this seems to be a sensible approach under the assumption that only what is perceived and understood can be used to construct and communicate a useful idea; and that we have to make those components perceivable, that are most helpful to get the user to discover what they consider to be *good ideas*, in particular hidden ones. This approach stands in contrast with empirical discovery of general facts without relating their possibly complex meaning for an individual or within a particular environment. [SHP+11]

3.5 Exploratory research

3.5.1 Setup

To better understand the problem situation (explorative thinking in a context of innovation) an ideation workshop at the CeRRI was observed. The workshop took place as part of another research project that explores the utility of crowd-sourced idea collections (idea-sparks) in ideation workshops, in particular their effectiveness in adding diverse points of view to the process and thereby improving outcome quality. The workshop tasked five creativity experts in a group setting to develop opportunities for innovation based on a new technology[20]. A task they all perform routinely in their day to day work. To aid them in this task they were provided with 200 crowd-sourced idea-sparks printed on index cards. All participants were familiar with the technology. The workshop took place around a large table in a conference room familiar to all participants. One of the participants also served as the moderator of the workshop. No technological aids were employed by participants during the workshop. The workshop was organized into three phases: For 30 minutes the participants familiarized themselves with the set of idea-sparks that was assigned to them; following that the idea-sparks were clustered by the whole group for another 30 minutes before engaging in group-ideation for the last 30 minutes. After the workshop each participant was interviewed for about 45 minutes on the influence of the idea-sparks on their process.

Figure 3.2: *Clustering during the ideation workshop observed as part of the explorative research.*

20 Transparent Conductive Oxides

3.5.2 Insights

The workshop and interviews were recorded and reviewed for this project in two rounds. The first round was a conceptually compressed transcript of interview responses based off the audio recordings. The transcript is included in appendix C. The second round reviewed the workshop recordings and further compressed the transcript into 18 concepts that appeared significantly in interviews or could be observed during the workshop. The concepts are included with detailed description in appendix A. Here included is an index of the concepts:

- Linear browsing
- Spatial arranging of cards
- Repeated reset
- Leaving and resuming
- Legitimacy of methods and tools
- Information processing rate
- Permission to break things
- Gaining and maintaining overview
- Embracing ambiguity (vagueness)
- Transformation of ideas into a persistent form
- Sharing and passing ideas
- Diversity of materials and other inputs
- Familiarization
- Convergent and divergent thinking
- Trial, error & prototyping
- Forming ideas from materials
- Connecting ideas to reality (with a plan)
- Playing the game (challenges, resistance, obstacles)
- Transitioning between ephemeral states of mind

This set of concepts, as it is, does not yet make it self-evident how to design a tool to improve the situation. The actual sentiment towards existing support tools among the participants in the workshop was this: *All tools helping ideation are terrible, therefore we don't use them. Why are they terrible? They do not fit the process.*

A fundamental requirement for any tool that wants to help in ideation then is making a tool that at does not impede the process. To do this, the process and its participants have to be understood better. A working model of the ideation process must be constructed.

3.6 A complex adaptive system model of innovation

To proceed with a pattern derived software design based on the observation in section 3.5, it is necessary to describe the model that informs the necessary design decisions and provides pointers to the specific aspects that must be validated in

the design's evaluation. Its purpose is to relate a set of observed and constructed concepts with each other, identify conflicts and offer some heuristic value that provides the grounding design rationale for a prototype implementation and subsequent evaluation through the lens of innovation as a complex adaptive system. It is also intended to sensitize for the dynamics that are supposed to exist in the context of ideation for innovation and to provide a concrete view of the problem that clearly points at the complexities involved, explains effects and suggests strategies.

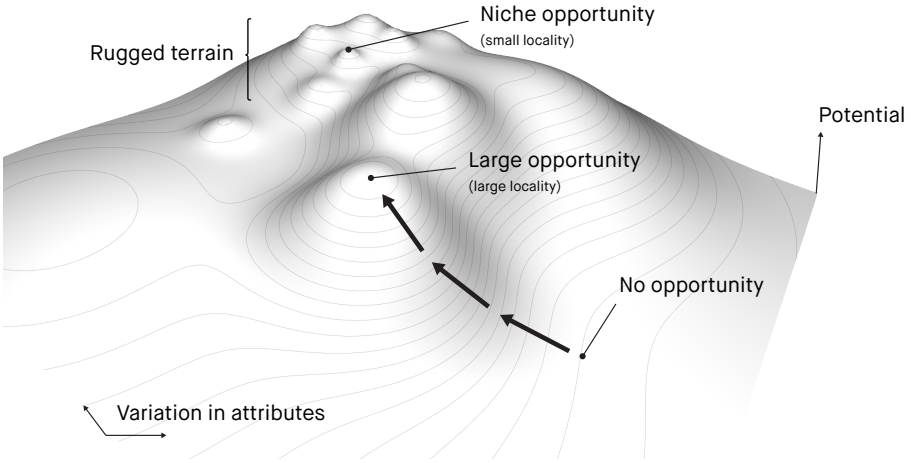

Figure 3.3: *Variation in the state of an object seen as a landscape of varying potentials based on fitness for a particular purpose in an environment.*

The model used here posits the idea that innovation[21] of artificial things is similar to the evolution of organisms as it is modeled by [Kau93]: To construct an analog, suppose artificial things are objects with a state; Then there exists a set of possible states immediately adjacent to their current state that are reachable by making one elemental change to one of its attributes. These possible states when so realized are more or less well adapted to the demands of the environment the object exists in and experience a higher or lower rate of adoption[22] as they are perceived as more or less useful by the environment. This model supposes that innovation, like evolution, progresses along the edge of what is possible under the constraints of a local environment, that no leaps are made and that therefore all innovation is dependent on a previous innovation that directly enabled it.

The scaling of this model is somewhat variable and of little help when applied at a micro level. However when viewing objects as organizations, laws, stories or social norms, the model reveals a certain heuristic value that can be used to guide the search for opportunities: *What changes to the attributes of an object are necessary to make it more useful? Are they possible? And how?* After all, in innovation of the artificial, each innovation has to be realized by someone and must therefore be discoverable. Furthermore the interpretation of utility (or fitness for

21 Innovation: An idea that is realized and has a significant effect on the world.
22 adoption rate would be reproductive success in organisms

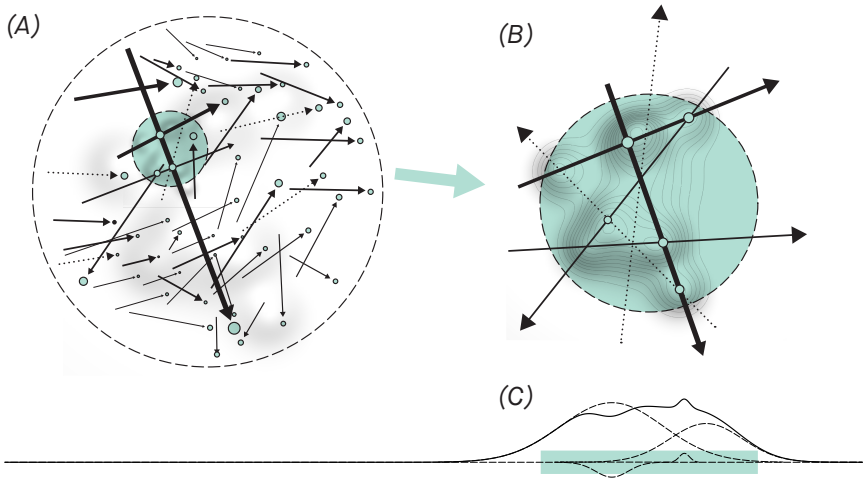

Figure 3.4: *(A) the influences in a problem space interpreted as an irregular web of influences. (B) The interaction of known and hidden influences in a part of the problem. (C) The potential utility that could be derived from an optimization of influences.*

a particular purpose) of a new object state can be viewed as a landscape whose features allow speculation about the contingencies of near term state transitions (figure 3.3). This idea of exploring adjacent possibilities is discussed at length in [Kau93] including the effects it creates that lead to complex adaptive systems and a description of an environment where it works particularly well, which is leveraged in section 4.1.1 in the prototype design.

Inspired by [Sim69], the objects mentioned above, can be seen as localities that exist in a hierarchical and near-decomposable relationship with various others[23]. They contain themselves systems and are influenced along various dimensions by larger ones from the outside. In an innovation process those would be become optimized to create systemic effects and begin to exert influence on their neighbors, illustrated in figure 3.4.

This model is subsequently referred to as Complex Adaptive Systems Model of Innovation (CASMI). Hereafter a number of references to CASMI are made where it visibly relates to design decisions and observations.

23 They form self contained units that influence each other only in aggregate

4 Prototype

Summarizing CASMI, the process of ideation for innovation is essentially a deliberate exploration of various yet unrealized configurations of existing elements that are reflected upon by the ideator on their contingent utility for solving a given problem. These elements each have their own history and systemic influence that is observed subjectively by the ideator who places them as landmarks in a mental landscape and imagines the geography between them. This landscape changes progressively, it gets refined, partially discarded, moved around, scaled, forgotten or emphasized as the journey through it progresses and new influences, facts and relationships are discovered. The ideators purpose is one of modeling a plausible landscape given a situation, visualizing and validating it as much as possible, before deriving a local map that guides the trial of those ideas that might actually work.

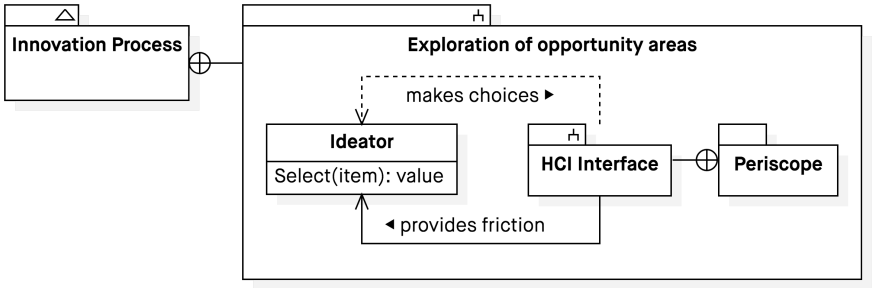

Figure 4.1: *The* HCI *between Periscope 0.5 and an ideator forms an agent for exploring opportunity areas.*

This chapter describes the design aspect of a information systems artifact that accompanies and supports such ideation. The implementation of its design is hereafter referred to as Periscope 0.5 and the pattern underlying its design is referred to as Stacktree.

First, the static requirements this system must satisfy are established. These are looked at from four points of view to cover the full picture of what influences the effectiveness the design: the conceptual needs of goal directed ideation (innovation), the user's needs in order to become a productive agent in the process of innovation and the organizational & technological boundaries a software system must relate to. Then follows a description of the effects the interaction with the design must have in order to become an effective agent in innovation. This

leads into a description of design principles by which such requirements can be realized before ultimately describing a specific design and its components.

4.1 Static requirements

Concepts that Periscope 0.5 must facilitate individually and at any point in time.

4.1.1 Conceptual needs

On the surface, the Periscope 0.5 is supposed to be a software that helps its user to think about problems in an explorative way and its design follows the needs of such an explorative thinker: An environment to capture the topology of a given problem space, where spontaneous configurations of elements can be discovered, where doors into adjacent possibilities [Kau93] can be recognized that open from these configurations, and where their effects on the world can be recorded for future reference. Such a process of selection works best when it happens at what Stuart Kauffman describes as *the edge of chaos* [Kau93], a landscape (environment) that has the following features:

1. A primordial soup of elements
2. A randomized force that may spontaneously create connections between building blocks.
3. An agent for the dissolution of non-useful connections.
4. Resistance to dissolution to provide opportunity for connections to form higher level connections into hierarchical building blocks.
5. A means to recognize and transport useful connections.

The effects of such an environment are modeled by CASMI in section 3.6 and variously discussed, exemplified and contrasted by [Sim69] (organization), [Kah11] (dynamics of cognitive processes), [Kau93] (origins of order), [Tal07] (origins of change), [Tal07]. In this project they serve as guide posts to aim the design process along a path that has at least the potential of reaching an effective conclusion.

4.1.2 User needs

Requirements derived from the abilities and limitations of people when tasked with creative problem solving. Derived from prior word (section 3.4.1) and exploratory research (section 3.5.2).

- Minimal cognitive energy spent on maintenance of method and environment.
- Ability to “just get started”; No plan needed to get started.

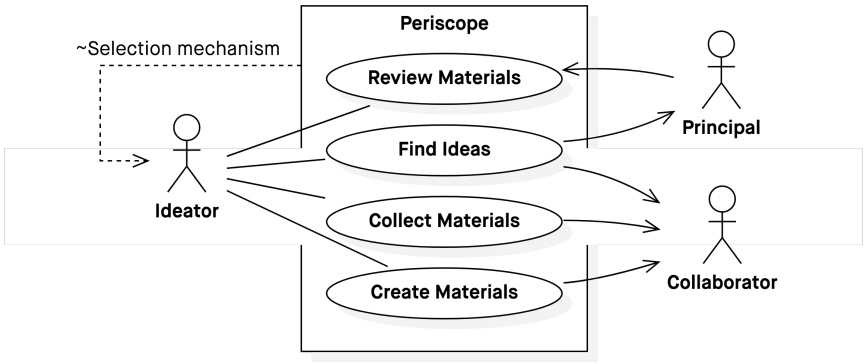

Figure 4.2: *The general use case underlying the design of the prototype. Of note is the dependency between Periscope 0.5 and the ideator reflecting the agency illustrated in figure 4.1.*

- Automatic, emergent structure that signifies progress, enables backtracking and keeps things scannable.
- Means to overcome mental blocks and maintain momentum.
- Diverse influences to help with outside-the-box thinking.
- Signifiers of state transitions, transparent behavior and mechanics.
- Transparency of method (it should appear as if there is no method).
- Simple but flexible structure; limited options that are not immediately relevant to the purpose of ideation.
- Integration of various media types: text, pictures, audio, video and drawings.
- Ability to re-purpose behaviors.
- Facilitation of group use.

4.1.3 Technological Boundaries

Requirements derived from the technologies that can be used to realize a prototype implementation. Also derived from the need to create testable scenarios for a prototype.

- Integration of local files (import)
- (deferred) Integration of physical objects (scan into local file via audio-/video-recording device), Uniform Resource Identifier (URI) (paste and load resource into local file)
- (deferred) Creation of assets in the application (mouse, keyboard, touch, stylus)

4.1.4 Workflow needs

Requirements derived from the organizational environment in which the design is to be used. These are the hypothetical situations of real world workflows and the concrete situation in which the prototype is to be evaluated.

- Saving and restoring sessions
- Importing assets (materials)
- Visible state transitions to allow live observation and live pair-/group-use
- (deferred) Exporting sessions for sharing, documenting, reporting and presenting work
- (deferred) Branching and merging of sessions
- (deferred) Extraction of partial sessions

4.2 Dynamical requirements

Concepts that must perceivably emerge as a result of a person interacting with Periscope 0.5. These are derived from CASMI in section 3.6. Their origin and nature is discussed at length in [Kau93] and [Sim69]. Their associations with observations from section 3.5.2 and prior work section 3.4.1 are listed.

4.2.1 Persistence and exaptation

Newly formed idea fragments must have an opportunity to be found and recognized as useful components in relation to existing ideas. Ideas must stay in view or remain (re-)discoverable for a while, they resist dissolution. Ideas may persist longer and transition into ever more complex configurations if they can be repurposed and carried as part of other configurations. They may become temporarily irrelevant and ignored by selection, thus surviving lazily in the background.

Related observations and concepts: *Taking notes — Clustering and classification — Reduction of complexity — Composition of ideas from shared fragments.*

4.2.2 Variation, error and noise

Faulty or incomplete information creates dissonance and in effect more creative solutions as this dissonance is overcome. Latent associations come into view while making sense of a perceived dissonance. Errors occurring in frequency much higher than the observed signal become noise and reduce the rate of information. Errors that aren't perceivable as noise are mistaken for information and can be opportunities for unexpected discoveries.

Related observations and concepts: *Diversity of creativity prompts and materials — Familiarization with and learning about the subject — Looking at examples — Strange, absurd, useless and misconceived inputs — Incomplete communication of ideas.*

4.2.3 Change

If a problem is viewed only from a certain point of view or if the problem is isolated and unable to respond to external influences, it is likely that its character is never fully revealed, certain aspects may remain hidden and actually variable properties appear static for lack of excitation.

Related observations and concepts: *— Walking around to change the view on present materials — Forgetting the current thought to make way for a new one — Shaking up thoughts that have become stuck.*

4.2.4 Constructive resistance

For the mind to become active it must be confronted with a perceived dissonance in perception that it can overcome. The more difficult the dissonance is to resolve, the more creative are the mind's attempts to explain it. At the same time resistance moderates noise and the rate of change, enabling persistence.

Related observations and concepts: *Constructing a game to play (willingly overcoming unnecessary obstacles). — Negotiating a contract between affordances and the role the user attributes to them — A variety of informal creativity methods — Physicality and its resistance to change — Finality of physical actions, unquestioned authority of physicality, energy cost of physical materials — Normative quality of ideas that have been communicated effectively — Opportunity for perceptual and adversarial learning — Opportunity for trial & error, self correction.*

4.2.5 Minimal cognitive load

Mental energy spent on maintenance of a method or environment is energy that cannot be spent productively. A new method or tool should cause no more overhead than the simplest existing alternative, relative to its value. Energy spent on handling frustration by a system and in recovery from stumbles during interaction with it are reducing engagement and consequently its impact on productivity.

Related observations and concepts: *Mental blocks and the sense of being stuck — Ability of a tool to capture the intent of a user — Ability of a tool to immediately adapt to shifting needs over the course of an ideation process: expansion vs. convergence, multiple threads of thought, note taking, offloading and sorting.*

4.2.6 Limited space

As items get added to a space, already present ones dictate where new ones may go. As the number of present items grows the space's overall resistance to change also increases, insertion of items between occupied locations becomes ever more difficult and prone to creating side effects that change the association between items. For example in a neatly laid out grid of sticky notes, insertion of a new note demands that a number of present cards have to be moved. The only alternative would be abandoning the semantics of the spatial arrangement and treating the collection of notes as a chronologically sorted list, where new notes are only appended. This is a property of a physical space which means that insertion necessarily displaces present items or happens at the fringes. In a virtual space this physical property of displacement can be changed arbitrarily. Items can be compressed with minimal resistance. Appending information at the edges of a space occupied by existing ones is however not how most thinking works. When thinking about something we refine our idea of it, or go on a winding journey from one idea to the next.

Related observations and concepts: *Arrangements of materials to track progress — Arrangements of materials to signify change — Loose relationships afforded proximity — Abandoning of large arrangements — Compression of arrangements into classes — Repeatedly starting fresh — Axial compression helps in identifying blind spots — Axial compression obscuring of axis perspectives — Small screens and low visual quality of projections.*

4.3 The Stacktree pattern

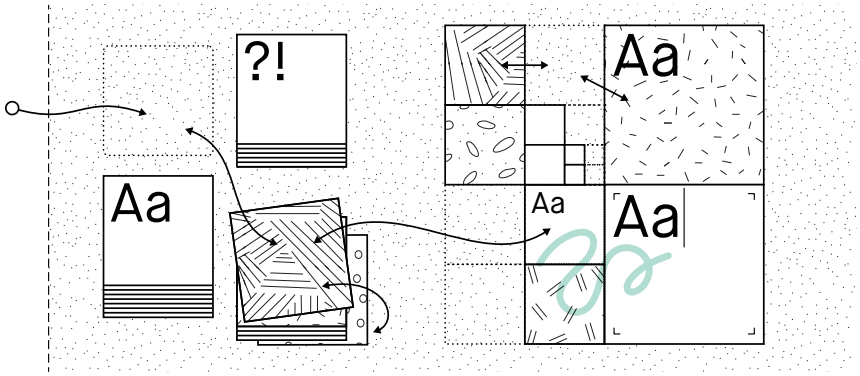

Figure 4.3: *The Stacktree pattern concept. Decks of cards and a quadtree that subdivides itself to accommodate "everything" in a constrained space.*

It was supposed at the start of the design that **a tree structure would likely exhibit properties that help in satisfying the requirements** above. A number of trees for spatial indexing were examined, among them quad-, Q-, Kd-, R- and

UB-tree. Among them a quadtree[1] offers the most intuitive and visually comprehensible structure that serves the scannable requirement well. The hypothesis going forward was that a quadtree is a good tradeoff between rigid and flexible.

In detail the following context specific properties and affordances of a quadtree were identified: (2) It offers a number of natural locations for controls and signifiers (four corners, four edges, center, frame, color, texture, vignette, front and back, relative size). (3) The center point between the four cards in the same region can serve as handle for transformations and commands targeting that region. This pattern works for all regions in the tree, enabling arbitrary group transformations and commands. (4) Card edges can be used to signify presence and absence of a boundary or border (map). (5) Texture can be used to typify a card, similarity between textures and colors can be used to indicate a commonality with and without adjacency. (6) Proximity can be used to indicate association. (7) Size can be used to signify dominance and a/symmetry of relationships. (8) Color and shading can be used to mark and highlight. (9) Corners can be used to place controls and signifiers out of the way of content. (10) Texture can be used for symbolism. (11) The square-based subdivision pattern favors no direction. (12) Subdivision is intuitive, self explanatory and somehow immediately suggests itself as useful. This specific character of a quadtree then suggested the Stacktree interaction pattern:

A Stacktree allows a user to lay out their ideas incrementally, create a landscape representing their conception of influences, opportunities, potential variations and promising intermediate steps toward their goal using scale, location, contrast, proximity, examples, symbolism and language on a plane subdivided automatically by a region quadtree.

Next to the tree, *stacks of cards* encapsulate the concept of input, output and work materials. Cards in stacks are generally filled with a variety of content. This content can have a range of sources, they can be self assembled, automatically generated or shared in a group. Instead of overview they provide a narrow focus, one item at a time. Their deck-of-cards metaphor gives them a predictable structure which suggests completeness and allows efficient processing of *all items* in a set. Additionally cards stacks are not singletons and can capture intent like temporary offloading, creating collections or reshuffling. Depending on card content, stacks can serve various purposes like providing directed inspiration, examples, challenges or additional information. They facilitate the aspects of variation, randomness, respite and help, all on demand.

Each region in the tree is represented by the cards it contains, these can contain a text or graphic and scale to fill the region bounds. Implicit in this structure and the behavior of the cards is a certain rigidity and visual order that captures

1 A quadtree subdivision of a space by points added to it, creates regions that appear as an ever more detailed description of a locality the closer the points are to each other. Resulting in a pattern similar to land division that gets progressively more subdivided as more differentiated land use occurs in close proximity, usually driven by the historic, societal, political and economic features of the region.

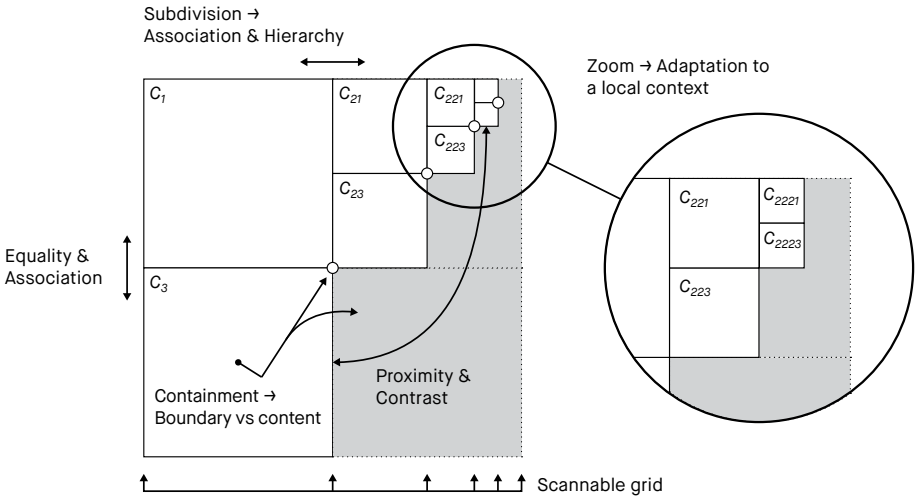

Figure 4.4: *The quadtree subdivision pattern affords expression of a number of relationships that can be used for capturing a landscape of ideas, influences and ideation progress in various zones. The abstraction of a literal map into a more abstract grid makes it scannable and simultaneously affords ad-hoc semantics for the loose relationships it creates visually. This simple and prominent structure encourages personal and shared interaction strategies to emerge.*

the needs and intentions of an ideator acting out the dynamics of exploration adjacent possibilities. The most important effect this pattern has, is that it maintains a scannable[2] structure while requiring minimal planning and maintenance. Figure 4.4 illustrates the affordances and geometric features inherent in quadtree subdivision.

Similar to the behavioral pattern of a subdividing tree, problem spaces — or problem landscapes — explored during ideation divide into finely differentiated hotspots where much opportunity and many competing ideas exist, and larger areas of new, unexplored, inaccessible, worthless or dangerous terrain. As ideation progresses, these areas become explored, a path is beaten through them, the space is subdivided mentally, features get discovered and added into it, maps of the region are drawn and tracks are left behind. These traces of previous visits can be used by later exploration for orientation, refinement of the landscape and finding places to settle where opportunities interact and webs of related ideas spring up. These dynamics are the same as that form the interactions in CASMI (section 3.6), however **now made tangible by translating their flexible theoretical shape into concrete and consistent behavior.** From this consistency certain self organizing patterns emerge that suggest structure, meaning and order. This malleable landscape of regions then becomes the material with which the ideator expresses the transformation of their mental state, their insights and speculations.

2 A *scannable* arrangement of items is easy to process repeatedly. Scannable arrangements have a clear and simple structure, a distinct silhouette and maintain their order [Ben09].

4.4 A Stacktree application design

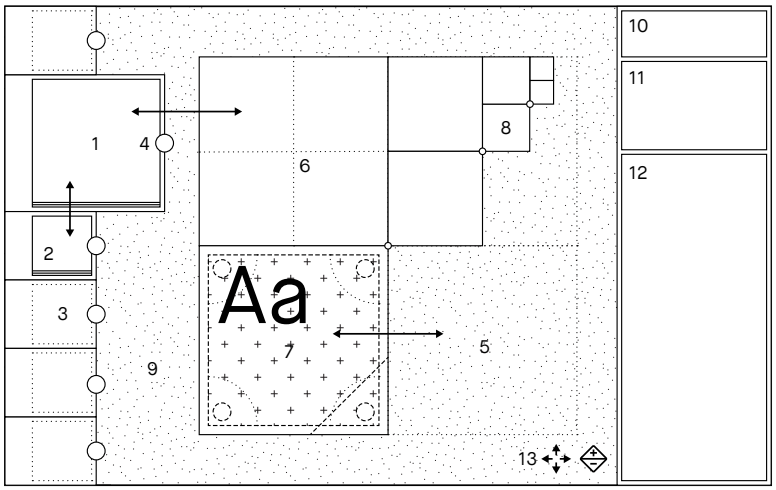

Figure 4.5: *A wireframe of the* GUI. *(1) highlighted card stack, (2) regular card stack, (3) empty card stack, (4) stack hotkey signifier, (5) empty region, (6) subdivision pattern of a region, (7) wireframe of a card covering a region, (8) subdivided regions in close proximity, (9) area outside the mapped region, (10) session information, (11) application commands, (12) context information and help, (13) pan- and zoomable view.*

The following sections discuss the adaptation of the Stacktree pattern into a prototype software design, identify the major parts, discuss their behavior and how they relate to what we want to achieve (better ideas). The design rationals for each individual decision are not explicitly given, as they would be exceedingly complex. Instead the rationals are taken as implicit based on the requirements described above and the design goals listed below. Further rationals become evident when examining the implementation in appendix B. It is inherent to this project that the opportunities afforded by technology, the concept of a possible application and the understanding of the possible deployment context of that application, influence each other. None of them are fixed in the attempt to optimize their systemic effect. Figure 4.5 illustrated the basic adaptation of the Stacktree into an application design. Figure 4.6, 4.7 and 4.8 show concrete examples of the Periscope 0.5 implementation of this design.

3 The content images seen in screenshots of Periscope 0.5 builds were provided to and chosen by a Periscope 0.5 prototype evaluation participant in solving a specific task they were given during evaluation and are included herein to accurately represent these choices in this particular context. The images are not part of the Periscope 0.5 application build, and are not distributed together with it. The choice to include copyrighted material is the user's own. The use of copyrighted materials during prototype evaluation was deemed to be necessary to provide a test scenario that is as representative of an actual scenario as possible.

Figure 4.6: *A typical screenshot of a zoomed-in view during an ideation session, showing various instances of cards used ad-hoc. Note: This screenshot contains copyrighted images*[3].

UI - A tree for overview

Tree design goals: (1) Enable efficient scanning. (2) Provide overview and (3) always enough space. (4) Allow loose expression relationships. (5) Intuitive, immediately explaining itself and inspiring decisions. (6) Be playful but also (7) serious and valuable. (8) Be the natural home of cards and maintain their utility. (9) Make multiple subdivisions legible. (10) Account for level of detail (LOD).

UI - Stacks for taking a walk

Stack design goals: (1) Capture the notion of a *card deck*, (2) serve as clipboard, temporary storage and trash can for cards, (3) capture the notion of *going for a walk*, (4) provide an endless supply of creativity prompts, examples, inspirations and starting points, (5) facilitate viewing a set of materials one by one (linear browsing), (7) allow scrolling through contents without skipping (8) allow fast forward without skipping, (9) capture the notion of shuffling a deck of cards, (10) effortless transfer of cards between stacks and the map, (11) large enough to comfortably view cards, (12) capture the notion of selecting or targeting a stack, (13) capture the notion of different types of stacks.

UI - Cards for taking note

Card design goals: (1) Create a widget that associates with a the concept of a card while not leading its interpretation into wrong assumptions about its capability[3].

3 During studies preliminary to this project (as part of [MSS+18]) it was found that disappointed expectations about the behavior of a widget were frustrating to the user

Figure 4.7: *An example of an initial state with imported assets supporting the ideation and a formulation of tasks and dimensions as starting points. Note: This screenshot contains copyrighted images*[3].

(2) Contain as much functionality in the card widget as possible without overloading its inherent capacity. (3) Continue the inherent structure from the tree (2 by 2 division) into cards as its natural terminator (leaf). (4) Allow associations of territory, boundaries, contrast. (5) Add minimal visual complexity through icons, frames and borders that over-emphasize a card's boundaries, which would prevent collages of cards to naturally blend into territories based on their color and texture. (6) Remain usable at scales between 100% to 20% (three or more levels of subdivision so 64 cards can be legibly viewed at once (assuming 1200 px vertical resolution). (7) communicate selected vs. unselected state (8) allow quasi direct manipulation of text (9). Provide triggers for adding & removal of attachments (video, audio, text, picture, symbol), bookmarking, tagging, drawing, duplication and deletion of cards.

Figure 4.9 illustrates the affordances and geometric features inherent in a square, minimally complex card design.

UI - Control scheme

Control design goals: (1) Fast and efficient control of the most used features. (2) An UI accessible for the target audience. Use of common shortcuts and metaphors similar to the most widely used tools the target audience likely knows and use of expected patterns (e.g. double click to edit). (3) Make other controls easy to

and hid affordances behind a misleading metaphor. For example representing cards specifically as sticky notes with drop-shadow and an earmark, immediately suggest that they can be moved. If such an interpretation is not immediately and categorically revealed as false the accessibility and perception of ease is take a significant hit as cognitive load increases while overcoming this dissonance.

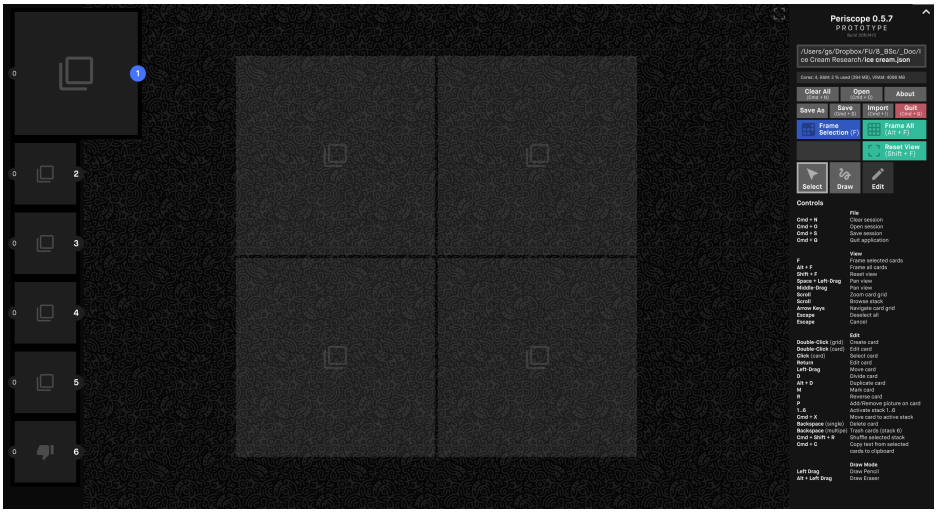

Figure 4.8: *The application after starting a new session.*

remember, provide a mouse-analog for all keyboards commands if possible. (4) Keep signifiers where the user is looking for them. (5) Avoid context menus (they are not well understood by many).

UI - Messages and signifiers

Signifier design goals: (1) Clear communication of application state, transitions and problems. (2) Use of animations to communicate the metaphor a widget uses instead of adding non-functional graphics indicating the metaphor yet containing no additional information or functionality. (3) Use of animations to embody interactions as much as possible and add weight to them right below the threshold where they become tedious (0.25 ms seem to work well). (4) Minimal visual complexity. (4) Use of messages to signify transitions and problems not inherently captured by any GUI component.

UI - Modes

Modal interaction design goals: (1) Modal interface for interactions that have mutually exclusive control needs (edit, select, draw). (2) Simple shortcuts and avoids input errors.

UI - Sidebar

The sidebar is in this project not integral to the testable behavior of Periscope 0.5. While its features are instrumental to constructing and administering test scenarios it consists of very standard features one would expect in any document based application.

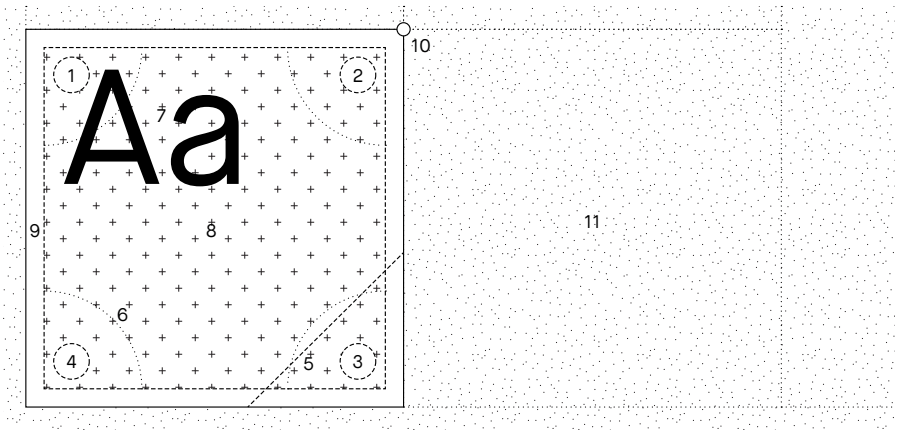

Figure 4.9: *A wireframe of the card widget. (1) delete card, (2) split/duplicate card, (3) toggle bookmark, (4) add/remove attachment, (5) bookmarked signifier, (6) card focus signifier, (7) card text display, (8) card attachment display and type area background, (9) card marked signifier, (10) parent node handle, (11) adjacent card placeholder.*

Transformation of the user

It is important not to confuse the intent of Periscope 0.5 with applications like Google Jam[4], Evernote[5], Pinterest[6], DEVONthink[7], Microsoft OneNote[8] . In abstract descriptions, certainly there is overlap, however Periscope 0.5 is a decidedly different application in that it aims to create a transformative experience for the user (much more like a game) rather than a collection, document or database. Certainly, Periscope 0.5 captures notes, but its value lies not in the note itself (as it is supposed by other applications) but in the particular behavior the cards inside Periscope 0.5 have relative to each other and the exploration this affords. Nevertheless the data created with Periscope 0.5 can — in principle — be transformed into various formats from hierarchical databases to structured text documents and posters, mostly though for reasons of embedding Periscope 0.5 seamlessly into established workflows rather than those exports being an end in themselves.

For the moment Periscope 0.5 primarily captures the spatial component of exploration and a decidedly forward motion. Chapter 6 contains an extensive list of features for continued development that extend the core pattern in a meaningful

4 Google Jam is a virtual whiteboard, available from `https://jam.google.com/` (July 30, 2019)

5 Evernote is a mobile app for note taking, available from `https://evernote.com/` (July 30, 2019)

6 Pinterest is a web and mobile application for discovering and collecting images, available from `https://pinterest.com/` (July 30, 2019)

7 DEVONtechnologies DEVONthink is an artificial intelligence (AI) supported note taking application, available from `https://devontechnologies.com/` (July 30, 2019)

8 Microsoft OneNote is a note taking and sharing application, available from `https://onenote.com` (July 30, 2019)

way. A wider deployment of Periscope 0.5 is also needed to discern the effects of the Stacktree on ideation outcomes in various organizations.

4.5 Implementation

4.5.1 Failure risks and conceptual architecture

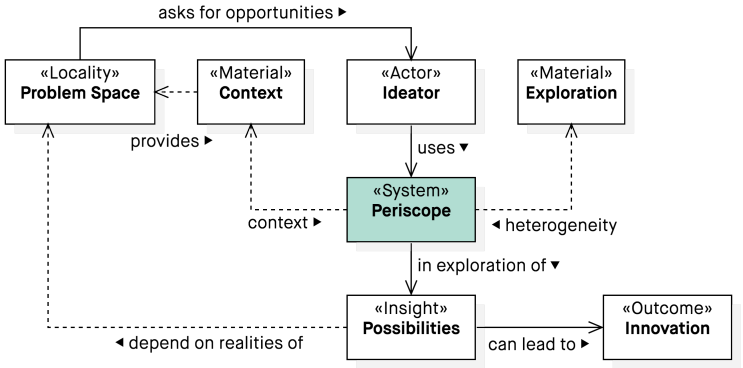

Figure 4.10: *The embedding of Periscope 0.5 into its context, reflecting the generalized use case that drove its design.*

Architecture, i.e. the *important things*, does not only become pertinent in the implementation phase of a project. It also influences much of its conceptual design, however in the design phase, the architecture does not stand at the beginning, nor is it the outcome, it emerges as the critical things that are discovered iteratively and incrementally over its course. But, design isn't just critical things, design's concern is with relationships that create systemic effects, which is a view different from architecture. Therefore it makes sense here to assume a decidedly architectural view of Periscope 0.5.

figure 4.10 shows the context embedding of the Periscope 0.5. Next to opportunities, which are subject of design, this context embedding also contains risks. Before making decisions that are difficult to change, it is sensible to think of architecture as *the things that deal with risks of system failure.* For this project these are functional as well as non-functional, some are systemically intertwined with each other and some are also part of the conceptual design (section 4.4). Table 4.1 contains a list of failure risks identified for this project that the implementation architecture is largely motivated by.

4.5.2 Technology selection

The fundamental paradigm in the design of Periscope 0.5 is one of creating behavioral entities that will be configured by the user to create complex implicit

Table 4.1: *Failure risks*

Cost & duration of development due to insufficient library support. → Use a mature technology stack and tooling.
Cost & duration of development due to lack of experience with the technologies chosen. → If possible use C#, Unity, .net, html5, css, Python, Gtk#, C.
Cost & duration of development due to necessary changes to architectural decisions. → Make architectural decisions as late as possible. Progressively constraining options as information becomes available.
Cost & duration of development due to fragmentation of responsibilities / over-engineering / feature creep. → Only build what is needed and sensible for the scope of this project based on these here failure risks.
Cost & duration of development due to critical bugs. → Build strong modularization into the implementation architecture (figure 4.11).
Not testable due to critical lack of visual performance/fidelity (no lag). → Use a ui framework with gpu support and a compiled language.
Not testable due to lack of portability. → Use an application server that runs on everyday PCs and deploys as an independent package.
No testable utility due to lack of suitable test participants. → Implement means to create and setup various test scenarios quickly
No testable utility due to lack of suitable test scenarios, i.e. test participants actually suspend disbelief and identify with the scenario. → Implement the concept of a serializable session that can be saved and restored to facilitate designing and using variable test scenarios.
No testable utility due to lack of suitable content for test scenarios. → Implement means to import media assets (text, images, others) into a session.
No testable utility due to data corruption and undefined behavior. → Simplify a session's state to be easy to validate, recover and reason about.
Diminished usability due to slow iteration of ui and ux. → Use composition and leverage the Unity Editor for configuration.
Diminished usability due to obtuse and inaccessible ui. → Use composition and leverage the Unity Editor for iterating designs. Internal test loop. Actually use the app.
No utility due to conceptual mistakes. → Understand the problem situation, its contingencies and develop a suitable model that informs the design (subject of this project).

interactions and effects through the user's eyes. Realizing this kind of interaction means, that a large part of the software would simulate an environment or world in which objects can be placed and manipulated. This world would have to have a variable viewport which would display objects of various sizes. The natural approach for modeling such a view would be one of a camera looking into a scene. Traditional GUI frameworks generally offer no concept of a camera, projection or zooming and would be difficult to adapt to one. A simpler approach lies in looking at game engines to serve as framework for developing a prototype. The advantage of this approach is that a game engine naturally builds on the concept

of a scene and is designed for creating the widest possible range of experiences, making as few assumptions as possible about how a user will interact with the finished application while still providing support for quickly building and iterating application designs with multi-disciplinary workflows. For this project Unity[9] was chosen as development platform. A detailed discussion of this choice is included in section B.1 and B.2.

4.5.3 Logical architecture

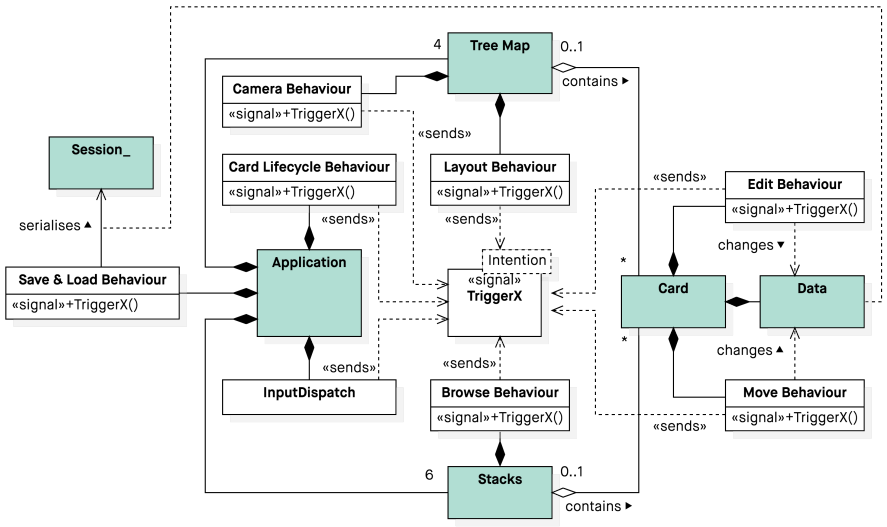

Figure 4.11: *A view of Periscope 0.5's architecture illustrating the decomposition into cards, stacks, tree and behavioral interaction through triggers.*

The components described in the Stacktree and Periscope 0.5 design above are represented in the implementation as agglomerations of Unity components. These components are organized in a part-whole hierarchy of Unity `GameObjects`. This hierarchy is used extensively in the implementation for maintaining a decoupled and cohesive structure. As a result the implementation architecture is not readily evident from the class structure of the implementation. Figure 4.11 illustrates the key Stacktree-modules and their conceptual associations in the implementation. The tree, stacks and cards are each treated as largely independent modules that define their own internal behavior and effects on objects contained in them. Among them they largely interact through their abstraction as rectangles in a vector space, represented by a Unity `RectTransform`-component on each `GameObject`. The control relation between these modules and the application's user is described by triggers that are variously captured and dispatched, depending on

9 Unity is a real-time development platform for hardware accelerated 3D, 2D, mixed reality (XR) visualization. It uses mono (an open source implementation of .NET) with C# for scripting and accessing the engine application programming interface (API). *Unity is available from* `https://unity.com/` *as of July 30, 2019*

context and the provisions the Unity Engine makes for them. On a conceptual level they are thought of as a separate module tying the stack, tree and card modules together. Ancillary to the modules that enable the Stacktree pattern is an application management module that represents the concept of a session, manages its state, makes it persistent and provides a framing typical of a document based application.

4.5.4 Implementation details

A considerably more detailed description of the Periscope 0.5 implementation is included in appendix B for reference.

4.5.5 Repository

The sources, builds and materials of the implementation are hosted in a Bitbucket repository and publicly accessible. Standalone builds of the prototype implementation for macOS and Windows are included. The repository can be accessed at the following URL:

`https://bitbucket.org/gerolds/periscope05/`

4.6 Evaluation

The evaluation of Periscope 0.5 in this project is qualitative. Its aim was to identify all good and bad effects that occurred in the experiment, treating each of them as equally valid and important and discovering their possible interactions. Since the population in the evaluation was so small (only five participants), it is important to also consider the particular participant in each experiment, their point of view, attitude and background in order to filter, or make note of various uncontrolled parameters where possible. To capture as many angles as possible, and normalize the collected data as much as possible in advance, experiments are carried out following a standard script with multiple phases: A demonstration to create a knowledge baseline, self directed exploration to create familiarity (observed), a task to prompt goal directed interaction (observed), a questionnaire to capture directly comparable responses on attitude and perception, an interview to capture the particular situation of the participant, their background, framing, opinions and insights and lastly a recorded observation of the participant capturing effects that were missed or omitted in the live setting. The evaluation script is included in appendix E.

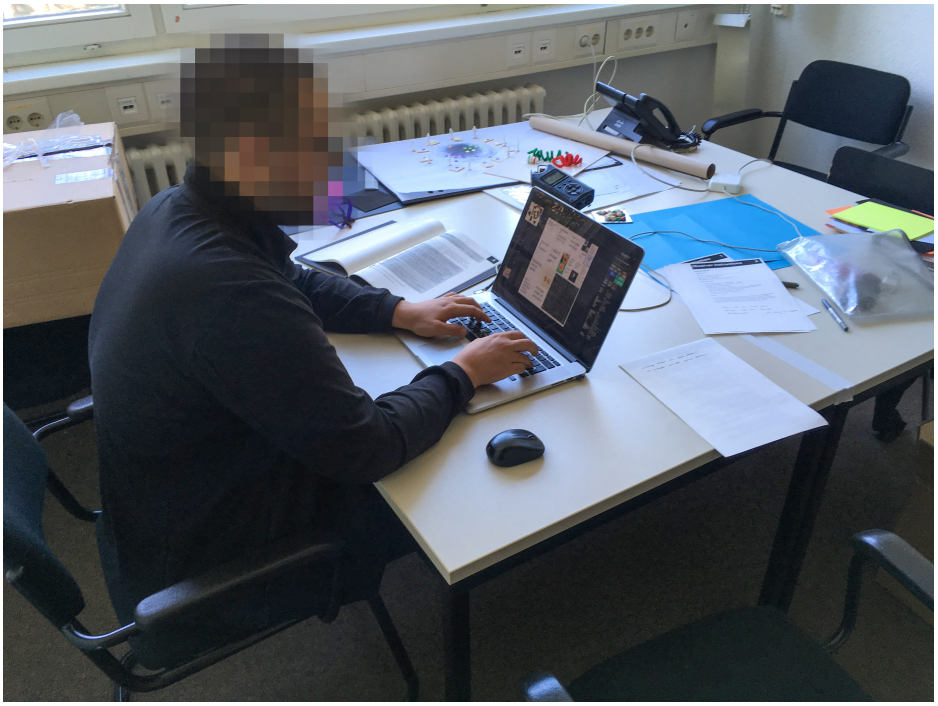

Figure 4.12: *The evaluation was conducted in situations similar to this, in a quiet office environment, on a notebook with mouse control. The notebook's internal camera and microphone together with a screen recording daemon were used to capture the participant's interactions*

4.6.1 Setup

The original plan was to evaluate the software with the same five persons that were participating in the exploratory research (section 3.5). However only two of them were available. The remaining three participants for the evaluation were selected based on their previous experience with design problems, the challenges of creatively solving tasks, methods supporting such tasks and basic theory that explains the dynamics at play in working on such tasks. This selection reflects also the target audience of the application that is supposed to feel comfortable with the assumptions it makes. Going into the evaluation, all participants were entirely ignorant about all aspects of the evaluation subject, the design of the prototype or its concept. All any of them knew was, that it somehow related to ideation and that it was software.

The evaluation was conducted in a place chosen by the participant based on the requirement that they would use this space if they were to spend an hour working on a difficult problem that requires creative problem solving. All participants chose their usual office space or a variation of it that removed potential interruptions.

The demonstration and test setup consisted of a table with two chairs side by side, one for the participant, one for the the administrator/ observer of the evaluation.

A 15″ notebook computer with an external mouse was placed in front of the participant. The notebook's built-in camera and microphone were used for recording the audio-visual feedback during demonstration, familiarization and task. For the interview phase only audio was recorded. The participant was given a printed evaluation protocol duplicating the verbal instruction given by the administrator and including various yes/no and Likert scale questions the participant would answer between phases.

The evaluation started with a scripted introduction and overview that was read to the participant explaining the format of the evaluation and their role in it and confirming their consent to be recorded and their data to be used for this evaluation.

4.6.2 Protocol

Phase I: Demonstration

The demonstration was a scripted performance of the administrator, explaining the purpose of the software and all controls, interaction patterns and opportunities the software provides that would be relevant later for phase III. The participant was encouraged to ask questions which were then answered until the participant was confident they understood the software.

Recording: audio, video, system-output, system-input
Duration: 5 minutes

Phase II: Familiarization

Immediately following the demonstration a tutorial session was loaded into the software and the participant was given control. This tutorial session included a number of cards that represented each of the points they were previously shown in the demonstration, aiding recall and inviting the participant to try them out themselves. Participants were again encouraged to asked questions, especially on controls they had forgotten. When it was noticed that the participant had issues with the tracking and scroll speed of the mouse, these were adjusted.

Recording: audio, video, system-output, system-input
Duration: 5 minutes

Phase III: Task

The participant was given a choice of three tasks they could now work on for fifteen minutes. They were encouraged to choose a task that they found interesting and felt comfortable working on. Participants were also given the option to select a task they would formulate themselves provided they already had a

design/creativity problem in mind that could be formulated with a specific outcome achievable in fifteen minutes and involving their particular area of expertise. After the task was chosen the participants were presented with a new session loaded into Periscope 0.5 that provided materials in support of their chosen task. Provided they selected their own task, they were only given a generic set of materials.

The materials provided in the session are a selection of images and text fragments taken from a set of creativity prompts from Brian Eno's *Oblique Strategies* [ES75], various quotations from Alan Fletcher's *The Art of Thinking Sideways* [Fleo1] and pictures published on *butdoesitfloat.com*[10]. The materials were selected quasi randomly[11]. Choices were made with the intent of providing an appealing, yet challenging set of images and texts that would be unexpected, visually attractive, unconventional (no stock photos), as different from each other as possible and covering the widest possible range of subjects. Visual tropes, phrases, symbolism and conventional beauty were avoided as much as possible. The complete set contains approximately 400 items of which 200 were selected at random to be included in the session.

Recording: audio, video, system-output, system-input
Duration: 15 minutes

Three participants selected the ice-cream task, two selected their own task.

Phase IV: Questionnaire

Based on the previous experience in phase III the participant is asked to complete 26-item usability assessment based on the USE Questionnaire [Luno1] with a few non-applicable questions omitted. The purpose of this assessment is to capture the immediate response after the task is completed in a way directly comparable between participants and to reference their interview answers with responses in the questionnaire. This then provides some indication of how much the usability and specific form of the prototype obscured the utility of the interface pattern that is actually the subject of the evaluation. This particular questionnaire was chosen as it seemed to serve this purpose adequately. The data obtained from the questionnaire is used only descriptively.

Recording: none
Duration: 5 minutes

10 `https://butdoesitfloat.com/`
11 similarity between items was deliberately prevented, at most two similar items were included.

Phase V: Interview

Immediately following the questionnaire a semi structured, fifteen minute interview is conducted to reflect on observations made previously during the task and to discuss the opportunities and problems concerning the software that the participant observed. Participants were asked to identify the most significant aspects of the software they liked and disliked. Then reflect on how the software influenced the outcome they achieved while working on their chosen problem and where they thought improvements could be made.

Recording: audio
Duration: 15 minutes

5 Results and discussion

In the previous chapters describing the steps taken in this project the results that stood at the end of each step are largely included at place where they occurred. The discussion on this chapter concerns itself with the results of the process as a whole.

5.1 Evaluation results

This review of the collected evaluation data is aimed at identifying the significant dissonance between design intent described in section 4.4 and the actual effects of the Periscope 0.5 implementation that could be observed. An exhaustive comparison of the design goals with the observations during the evaluation has been forgone, as the evaluation data obtainable under the given organizational constraints data does not permit such analysis with accuracy. Instead Periscope 0.5 and Stacktree as a whole are examined and traces to individual design decisions are established where they appear. A compressed versions of the interview responses is included in appendix D. The original recordings of the evaluation sessions have been deleted in accordance with the conditions of consent given by evaluation participants[2] and data protection laws.

5.1.1 Process evaluation

In figure 5.2 the results of the usability assessment show a distribution leaning towards agreement with the questionnaire statements, indicating a generally positive bias of the participants towards the software. When correcting for it by raising the floor of the scale from 1 to 3, it still appears the software is perceived as decidedly useful, pleasant, fun, easy to use, meeting the needs of its users and easy to learn and only slightly cumbersome, complex, somewhat unsafe and unpredictable. Based on these results it can be assumed that the usability of Periscope 0.5 did not significantly obscure the utility of the Stacktree pattern in this particular population. It seems the participants understood the efficacy of the design and found ways to make it work for them. Therefore the observations made during the evaluation should be attributable rather more to the Stacktree pattern than its implementation.

1

2 Appendix E includes the consent form used for the evaluation.

Figure 5.1: *The course of a typical session during evaluation. Note: This screenshot contains copyrighted images*[1].

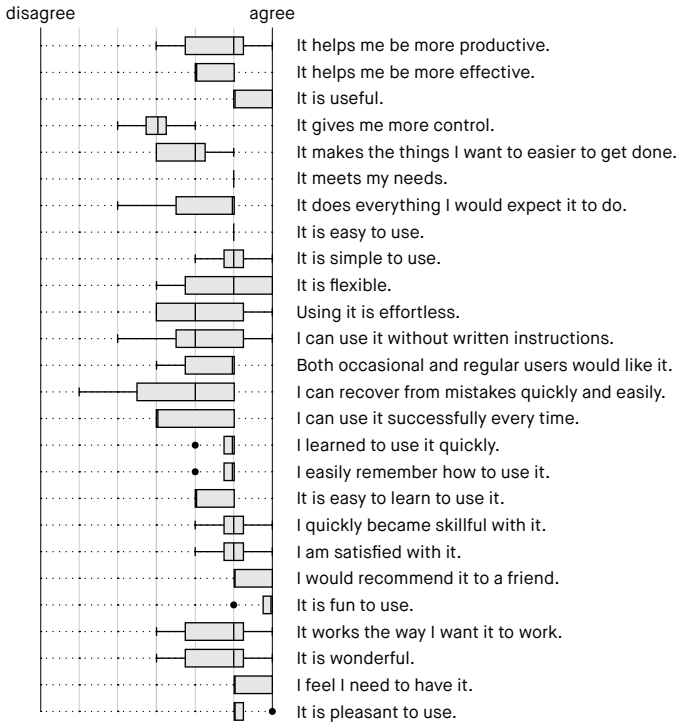

Figure 5.2: *The five participants' USE-Questionnaire responses on a 7-step Likert scale. Due to the small underlying population this analysis is purely descriptive.*

Participants responded to the demonstration and familiarization phases with a slight rise in self assessed confidence toward their being able to use Periscope 0.5 effectively. Overall though it seems that they were unable to make a clear distinction in their confidence assessment. In the interview later it was variously mentioned that the interface seemed complex or intimidating at first. So a possibly greater understanding of the software was still counterbalanced by an awareness of its superficial complexity.

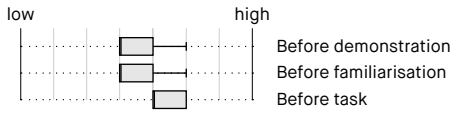

Figure 5.3: *The five participants' self assessment toward their confidence of using Periscope 0.5 productively. Due to the small underlying population this analysis is purely descriptive.*

5.1.2 Open observations

Participants' responses immediately after the evaluation were decidedly positive. They ranged from mild to enthusiastic. The Stacktree was described as interesting, easy to understand, easy to learn and unexpectedly flexible. Its premier quality

was described as challenging decisions on what is important and helpful to get started.

All participants learned to use the Periscope 0.5 productively after 5 to 20 minutes. Time to learn was strongly correlated with their previous experience using 2D/3D design- or layout-tools. The ability to arrange cards and by extension ideas, notes and pictures was welcomed and immediately used purposefully by all participants. All participants described Periscope 0.5 as daunting prior to the demonstration phase. Once they had used it in the familiarization and task phases, their sentiment changed entirely to "actually simple to use".

All participants selected a task that they found interesting (their own assessment) and proceeded with the task immediately. Only minor help was required in handling the Periscope 0.5 interface. It was observed that the tasks were approached in very different ways by each participant, reflecting their personal method of dealing with such tasks. It was variously noted that the exact framing of the tasks, that is, the specificity of the desired outcome made it easy to get going quickly.

Cognitive load:
Multiple participants' interview responses suggest that the Stacktree minimizes the cognitive energy that has to be dedicated to process setup and maintenance[3], reducing fatigue and resistance to engage in exploration of subjects. It also reduces the waste of cognitive energy spent on dealing with the frustration of mental blocks by offering useful inspiration, opportunity for reminiscing on recent results, casual browsing and ad-hoc changes of perspective.

Subject research:
All participants were more or less productive in their ideation during the task phase depending on their ability to ideate ad-hoc, without requiring prior research of external opinions and facts about the task subject. This seems to be unrelated to the Stacktree pattern and Periscope 0.5. Although it might be an opportunity for future development to make provisions for this need. Multiple participants suggested that material stacks providing specific background information, examples and facts about the task subject would be helpful to them.

Own materials:
All participants variously suggested that adding their own materials to stacks would be required if they were to use the software in their own work. Periscope 0.5 already supports this.

Drawing:
Most participants suggested that they usually want to capture some of their ideas through drawings and they would want to include them into the Stacktree or draw directly on cards or anywhere. The idea of drawing independently of cards, possibly for making meta-annotations and adding visual cues not immediately available by the Stacktree was welcomed. The Periscope 0.5 already supports drawing on cards.

3 requiring minimal planning and self control

Exporting work:
Some participants expressed a wish or need to be able to export the work they have done in the Stacktree pattern into other formats. Structured text documents and images capturing the layout were suggested.

Overall visual appearance:
The overall visual appearance of Periscope 0.5 was mentioned in three different ways. To one participant it appeared to suggest creativity in an overbearingly explicit way (attributed to the background design), to another it appeared decidedly uncreative and formal (Highlight color blue), and to yet another it immediately made them feel welcome (again attributed to the background design). This suggest that there might exist some value in making the appearance of a specific Stacktree pattern implementation configurable to make all users feel welcome.

Card highlighting:
Regarding the design of cards, the selection signifier (coloring the entire card medium blue) was described as making the content of a card hard to see. A frame instead of coloring the entire card was suggested. The frame approach was tried during development and discarded because of the insufficient contrast it provides across different card sizes. While the problem is significant, it is not critical. A solution remains elusive.

Content:
All participants mentioned the provided material stacks to be a significant influence on the ideas they produced during the task.

Accessibility:
For participants who had previous experience with layout and CAD software or third person 3D games, Periscope 0.5 was immediately accessible. Users that are not familiar with the concept of a camera, i.e. a pan & zoom views, have found their initial fifteen minutes considerably less productive.

Joy:
Participants repeatedly described Periscope 0.5 as joyful and easy. During the task most of them seemed to enjoy themselves and positive responses (through voice inflection and facial expression) could be observed. This in itself might be a factor that improves outcome through repeated positive affect [LKD05].

5.1.3 Axial observations

Model congruence:
All participants fully understood the Stacktree pattern, derived strategies for their own purposes from it and could immediately begin using it without frustration. The ubiquitous initial behavior was to make some sort of notes that would establish the primary task goals and distribute them to cover most of the space provided by the tree map. Following that the dominant behavior seemed to aim at capturing the main influences acting on the problem situation that was subject to the

task. This behavior closely matches the concept of a local space, influences and distribution of focus points (knots) of CASMI.

Stacks:
All participants used the stacks component of the Stacktree pattern for browsing the material stacks included in the pre-configured evaluation session. Upon inquiry as to why only for that purpose, all participants seemed to have encountered no other use for them and speculated that this was because of the task-session's short duration of 15 minutes. Most of them supposed that during extended use they might become useful not only as input channel (different types of material stacks) but also for other purposes, no specific uses were mentioned. The original design intention of stacks was to replicate the idea of a deck of cards in the hand of a user and facilitate the behaviors commonly associated with them. This connection was not made by participants which suggests that the UX of stacks requires improvement. Nevertheless stacks were used extensively to browse materials. It was mentioned and repeatedly observed that stack browsing was used to "stay in the flow" and to quickly arrive at a range of initial ideas.

Materials:
Participants variously described the content selection offered as materials in the task session to be very "nice" and actually helpful in getting started. In most instances the materials provided foundations and contexts for the ideas generated during the task. In particular the ability to modify materials was welcomed. Materials were used in various ways by participants, ranging from selecting favorites in a catalog to using them as building blocks and signifiers, or deconstructing them into wholly new things.

Cards:
The UX of cards was not mentioned in particular. This indicates they behaved as expected, which suggests their design is effective. This contrasts with the design of cards in [MSS+18] where their design was a major source of confusion and criticism.

Tree:
Participants welcomed the "rigid yet flexible" quadtree structure imposed by the Stacktree. They attributed it with the ability to maintain just enough order to be useful while not being oppressive. The consistent subdivision pattern that divides region into four equal sub-regions creates a strong emphasis on powers of the number four. This makes it awkward to categorize items that do not come in fours. This was noticed by all participants. Most did not feel that the number itself or the geometry resulting from it affected their outcome quality. When reflecting on problems that might arise from the Stacktree, ways to solve these problems seemed to suggest themselves to participants immediately. Some of them were hacks[4] of the system that would circumvent its usefulness as adversary to conventional thinking. The observed hacking suggests that the tool can

4 A *hack* here refers to the clever use of tools, objects or systems for purposes other than for what they were intended.

be easily misunderstood. It seems likely that this misunderstanding is caused by habituated expectations of what an application is supposed to do[5].

Persistence and Exaptation:
The tree captures ideas immediately through new cards, as more cards get added, the local area where new cards originate gets crowded, pushing other cards in the background, this forms a sort of landscape which is indicative fo the work and history that went into it. Concise and large cards receive more attention, this means unexplored areas and compact ideas get more attention. Deliberate action is required to pull good ideas up, make them appear larger. There is a constant drive to simplify to make things more visible.

Variation, Error and Noise
Materials in stacks are quite unpredictable. Materials rarely match anything in particular, requiring them to be used in a metaphorical, suggestive, evocative, associative way. Participants welcomed this quality of materials. From participants behavior during task sessions, it seems they are strongly attracted by the ability to browse through a stack of interesting materials.

Change
There is always something new to look at through material stacks. This purpose of stacks was understood by most participants and used extensively. Absent the use material stacks the deliberation periods during the task session seemed decidedly ponderous. Participants exhibited behavior that strongly suggest that stacks were used to overcome loss creative momentum and preventing blocks. Interview responses were also indicative of this behavior.

Constructive Resistance / Dissonance
The Stacktree challenges decisions on where to put something, what is important, how to formulate it, how to represent it. It produces chance associations through the subdivision geometry and its constraints.

A solution to the space problem
It seems that the Stacktree solves the space problem. It was not mentioned by participants.

5.1.4 Action falsification

It seems fairly certain that the Stacktree pattern is at the very least inconsequential to outcome (no negative effect). Whether or not the pattern actually does improve outcome is difficult to judge from the limited data collected so far. At least in the small initial population of five what has evaluated Periscope 0.5 for fifteen minutes, four mentioned at least one instance where they felt that *the pattern lead them to a novel idea that positively surprised them.* How substantial these ideas actually are, is unknown. Nevertheless this is a promising result that encourages

5 Document preparation, construction drafting, presentation design, etc.

further study. Given that Periscope 0.5 did not obscure the efficacy of the underlying interface design pattern, it seems likely that the initial observations of an ideation workshop, CASMI and design of the pattern are congruent up to this point. No clear evidence of misconceptions was found, which is surprising and should be checked in subsequent assessments of the pattern.

5.2 Periscope 0.5 and Stacktree limitations

The pattern requires **familiarity with the concept of creativity**, some theoretical understanding of it and exposure to the most common methods used for ideation, sense- and decision-making in order to understand the value it can provide, which is necessary for users to engage with the design and accepts the challenges it offers[6]. This directly relates to the question of design legitimacy as a central requirement for its effectiveness.

The pattern depends on **quality content** to be effective, it serves as a catalyst for it. If the content is badly curated or divorced from the purpose of its use, outcome quality is actively diminished[7]. If no content is available, it has to be constructed, which requires significant effort. A study of supplementary patterns that conceptualize and explain the efficacy of various selections of materials are required.

Periscope 0.5 is significantly more accessible to users with previous experience in using production tools whose main interface relies on direct camera control, like painting, layout and CAD software. This indicates that **low agility** with pointing devices might obscure the usefulness of the pattern. This might also be an indication that an adaptation of the pattern to alternative input methods like touchscreen, gamepad or trackpoint are more or less efficient.

Drawing is an integral part of explorative thinking for many. Integrating such functionality in a useful way requires non-standard hardware devices which are comparatively expensive (tablet, stylus) and potentially cumbersome, somewhat limiting the utility of the pattern absent these. Additionally the mediation of drawing through a singular tablet device is a generalization of the many forms that drawing takes in the process of creative and explorative thinking. And as generalizations go, the particulars that make an instance of it actually effective, are lost. This is not to say that mediating drawing in a tablet device is impossible, just that it inherently affords different things than drawing on paper does[8]. Exactly how

6 This was clearly evident during testing of preliminary prototypes, i.e. [MSS+18] where placeholder materials and a purely random selection of test participants was used.

7 Prototypes created in support of [MSS+18] have shown this.

8 Compare, for example drawing on index cards, placed on a table shared by a group of five, with each of them drawing on simulated index cards on their personal tablet device. While the drawing itself is likely quite the same, the cards all exist in a separate world only visible through a small window of limited fidelity. This makes many interactions with the drawing awkward or tedious which would be natural and easy with real index cards. These natural interaction seem to be of such importance that despite honest attempts to make digital drawing work for ideation, there seems to be

the desire to draw on paper during ideation is translatable to a screen medium requires further study. Digital drawing has in professional production contexts largely replaced analog methods. It offers speed, faster iteration and branching which are overriding all benefits of analog methods for economic reasons alone. These benefits however do not bear the needs of drawing for ideation.

Screen sizes are limited and sharing an ideation session tied to a **screen representation** limits its potential for group settings. Periscope 0.5 implementation supports small group sessions by clearly signifying state changes through animations, large projections however suffer from poor visual quality and input being bound to a singular device.

Ideas that cannot be translated into less than 250 words or depend on a precise record of symbolic relationships cannot be represented by the pattern. The pattern is probably useless for thinking about mathematical concepts (i.e. formal relationships) and exact designs (i.e layouts, shapes, assemblies). Deferring the detailing to regular production tools and importing variations back into the Stacktree for continued ideation seems possible, maybe useful, if the tedium this creates can be removed.

Complex sessions can appear confusing and incomprehensible, limiting their use as a medium of communication with others without deliberate effort by the author to make them readable. Making them readable will likely have an unwanted side effect of turning the implementation into a production tool, establishing misleading signifiers that distract from the actual purpose of the pattern (accompanying explorative thinking). With continued use of the Stacktree pattern, standard structures (patterns) might emerge that mitigate this limitation somewhat.

5.3 Comparing Stacktree to other patterns and concepts

Versus spatial arrangement of items *without* automatic layout

The tree map has a persistent quality: Changes happen at any time to only two cards (the subject card and its immediate neighbor inside the same region). In a layout with fixed-size items, inserting an item into an existing structure introduces layout artifacts like gaps, misalignment, undue emphasis or shifts in proximity, that can easily affect the whole arrangement. While these effects may capture an adversarial quality their legitimacy is questionable and they are oppressive in their consistent negative effect rippling through the entire scene. They rarely create useful prompts for reflection (one can not reflect on n^2 pairwise changes each time something gets added to the scene).

Versus spatial arrangement of items *with* automatic layout

The Stacktree pattern specifically deals with the issues arising from unstructured

yet no implementation that has captured it in a way that improves meaningfully upon pen & paper.

and automatic layout. Unstructured layouts require the user to manage structure manually which requires effort spent on planning and maintenance. Lack of structure leads to arrangements that are difficult to scan. Automatically structured layouts usually change the position of items making them useless for using spatial arrangement for expressing relationships and keeping track of items. The Stacktree automates the subdivision of space automatically and limits changes to the immediate vicinity of an item. This means, changes occur only where the user is focusing their attention anyway and are small in number (two), thereby easy to track.

Versus Mind-mapping

Example: XMind[9]

The Stacktree captures the most helpful aspect (progressive refinement, class choice) and avoids the negative aspects (rigidity, tedium, dogmatic method).

Versus Whiteboards

Examples: Google Jam, Ideaflip[10]

A generalization of a whiteboards offers no inherent structure, it is therefore hard to scan and gets confusing quickly. They also offer no constructive constraints or friction, which can be interpreted as their virtue. Usually they offer multiple pages to overcome clutter. They are most useful as a collaborative live sketching tool. The Stacktree is more focused towards containing ideas in chunks and subdividing those to refine ideas. Visually the result of a whiteboard session is a stack of posters, the result of a Stacktree session is a collage of cards. Both whiteboards and the Stacktree can be adapted to be useful for sharing knowledge and tracking progress, albeit for different outcomes.

Versus Commonplace Books and Note-Taking

Examples: Notes[11], OneNote, DEVONthink

They capture lists of things (ideas), chronologically as they occur. Sort of a database. In particular DEVONthink aims to cross-reference items automatically. Their value to ideation appears when old notes are re-read and associate with a problem at hand. A similar concept is represented in Periscope 0.5 by stacks and also inherent in the affordances created by the tree map. Integration between a commonplace book and Periscope 0.5 might be very valuable. A commonplace book captures ideas by inserting them as new items to the existing list. In Periscope 0.5 this would be equivalent to having one large session (possible). It takes a while before a commonplace book becomes effective as notes have to be collected over time. Periscope 0.5 shortcuts this somewhat by offering random fragments through material stacks. Information gathering software is often feature rich and provides many tools to create, edit, share and organize notes, often departing from the simplicity and constraints an actual book has.

9 XMind is a brainstorming and mind mapping tool, available from `https://xmind.net` (July 30, 2019)

10 Ideaflip is a virtual whiteboard, available from `https://ideaflip.com/(July30,2019)` (July 30, 2019)

11 Apple Notes is a note taking application, available as part of macOS, `https://www.apple.com/macos` (July 30, 2019)

Versus Production Tools

Examples: InDesign[12], PowerPoint[13], Latex[14]

Production tools pose questions relating mainly to the needs of the software itself or the medium it targets. Often these are quite technical and abundant, overwhelming the user that just wants to capture ideas, not aiming at production results. At the same time the fidelity these tools make available during ideation can be relevant if the divide between having and expressing an idea is minimal[15].

Versus Authoring Tools

Examples: Scrivener[16], Final Draft [17], Unity Editor

These tools focus on capturing all types of media, adding metadata, classifying and cross-referencing items and offering search and filters to re-discover them. As preproduction tools, they target various media channels and offer tools that aid drafting, planning and iterating for media in that channel. Scrivener and final draft, both writing tools, for example offer various breakdowns of the items in a project that allow deconstruction and rearrangement of the narrative in development. The outcome of such companions if often a manuscript or blueprint of some sort, which is in itself not yet fully specifying the final product. Therefore they are hybrid between a production and a design tool offering the best of both worlds for a well specified intermediate task. Compared to Periscope 0.5 they shine in certain scenarios, but lack the flexibility of the Stacktree pattern. It is likely that they would be complemented rather well by a Stacktree implementation.

Versus Idea- and Knowledge-Management Software

Example: Spigit[18]

These are primarily data sourcing and analysis solutions that promise to provide insights based on *collective intelligence*. They are generally offered as a enterprise solution and seem to offer no innovation in terms of HCI. In contrast, Periscope 0.5 is a specific HCI interface aimed at individuals and small groups to improve their personal ideation outcomes.

12 Adobe InDesign is a desktop publishing software, available from `https://adobe.com` (July 30, 2019)

13 Microsoft Power Point is a presentation preparation software, available from `https://microsoft.com` (July 30, 2019)

14 Latex is a document preparation system. [Lam86]

15 E.g. logo design and video editing are examples that benefit much from the immediacy and fidelity offered by production tools while painting and sculpting sculpting require planned and deliberate action that are explored via sketching.

16 Literature & Latte Scrivener is a word processor and scrapbook to aid in research and manuscript writing, available from `https://literatureandlatte.com` (July 30, 2019)

17 Final Draft is a word processor for writing and formatting screenplays, available from `https://finaldraft.com` (July 30, 2019)

18 Plainview Spigit's innovation management platform enables organizations to crowdsource breakthrough ideas from the employees, partners and customers that know their business best. *Quoted from* `https://www.spigit.com`, *retrieved July 30, 2019*

6 Conclusion

6.1 Project summary

This project has set out to establish a design pattern for software applications that can improve outcomes of explorative thinking. On this subject in general and particularly in the context of innovation, an instance of a typical situation where explorative thinking takes place was observed. This observation was found insufficient for informing the design and construction of an application design that is a complement or improvement to existing methods and available tools. To understand the generalized dynamics of the observed situation, a conceptual model of goal directed ideation (CASMI) was developed, framing it as a complex adaptive system. The properties of an environment that fosters such ideation (innovation) were established. Based on CASMI and reflecting the observations, a specific design pattern was developed (Stacktree). The Stacktree pattern captures the essential properties of an environment conductive to innovation. Its components, their design goals and inherent, as well as adjunct affordances, were described in detail. For the purpose of evaluating the efficacy of the Stacktree pattern a prototype (Periscope 0.5) was implemented. Requirements of the design were translated into architectural decisions based on failure risks. The technologies used in construction of Periscope 0.5 were discussed and details of the implementation pertaining to the realization of the Stacktree pattern were described in detail. Following the completion of the implementation, a protocol for its qualitative evaluation was developed and administered to a group of five experts in creative problem solving. The results of this evaluation were analyzed and compared to the original observations, model and design goals.

6.2 Results summary

Overall the Periscope 0.5 implementation has proven to accompany explorative thinking in a constructive, intuitive and joyful way.

The usefulness of the pattern outside constructed test scenarios requires further study. The nature of an ideation context however is always somewhat constructed and objectivity is rather a result of the process than its starting point. As such it is likely that the effects observed in chapter 5 translate to real world deployment, not least because the pattern seems to be engaging and promising a joyful experience. What remains to be seen is, how the pattern works in longer and continued sessions involving more materials.

The positive affect created in users by this response has significant potential of improving outcome [LKD05].

6.3 Encountered problems

The main problem encountered in this project was the **disinterest of the target audience of the Periscope 0.5 prototype to participate in its evaluation**. The framing of this project as a bachelor's thesis project therefore has not allowed me to conduct the extensive research that would actually be required to comment conclusively on the usefulness of action research as the to-be-disturbed system (ideation at CeRRI) became elusive to scheduling.

The **evaluation method** used in this project has exhausted its usefulness quickly. It did not identify any critical problems or significant opportunities for improvement. Over the long term it might be more productive to deploy the prototype to a relatively large and interested group of alpha-testers[1] that provide unstructured, ad-hoc feedback on their experiences and supply usage metrics that are automatically collected by the prototype.

Incidental to **scientifically framing invention of a useful artifact**, there existed a notable conflict of process between actual invention and the expectancy of deductive reasoning that leads to invention. It turns out, they are very different. It seems almost like deduction only exists in retrospect: Invention has the inherent property of making large speculative leaps, not all traceable to peer reviewed and published research. Some of them are just perceptual learning and pattern matching that was trained over years of broad observation, drawing from an absurd combination of concepts that are seemingly unrelated. Capturing these in a way that makes them traceable by a disinterested reader and creating a legitimate argument for them, quickly ends up in **prohibitively verbose** descriptions of *everything*.

6.4 Opportunities for improvement

The Stacktree pattern appears to be sound and somewhat effective. The Periscope 0.5 prototype is largely complete and allows exhaustive further study of the pattern. Within this projects only the basic utility of the pattern for an individual in a constructed setting could be tested. A sensible next step would be a wider deployment of Periscope 0.5 to discern the effects of the Stacktree on ideation outcomes in various organizations. Evaluation of such deployment could be done by automatically collecting usage metrics and conducting interviews on an group level. To make such a deployment viable the implementation must not necessarily be stable but usable in a professional context. A set of sensible features adding to

1 A group of potential users that evaluate the prototype in as close to real deployment conditions as possible.

usability and improving workflow integration could be added to Periscope 0.5. Following is a list these.

Features improving productivity and workflow integration:

- Undo
- An option to import/export a session from/into a folder structure of text files and images
- Option to export to a flat document (outline)
- Portable Network Graphics (PNG) and Portable Document Format (PDF) export.
- Text export.
- Performance: differential updates to the tree layout (required for performance with more than 200 cards).
- Performance: LOD mechanism, making tiny cards cheaper to draw/update.
- Checking the robustness, especially under stress for the tree layout engine.
- Gracefully recover data from corrupted session files.
- Additional means to extract data from text files.
- Snapshots (maybe the same mechanism that provides undo)
- Save drawings
- Multiple sessions (tabs)
- Merge two sessions (to make sharing and iterative work easier)
- Send/ or export a quadrant (e.g. when done, or no longer needed without destroying it.) into a new session
- Send or export a selection of cards into a new session
- Allow line breaks in cells (could be detrimental to outcome by encouraging too much text in cards)

Usability and UX features:

- Improve the design and implementation of stacks to make them more useful.
- A way to color cards.
- A way to bump (upvote) cards.
- Keyword search to quickly find and navigate to cards based on recalled words. This would encourage the emergent use of hashtags. There exists a potential that search would be detrimental to outcome.
- Markdown to format a card's text
- Paste URIS and auto-import the target if compatible.
- operating system (OS) drag receiver.
- Auto update assets when their source file changes (allows file system based editing of session content and external scripting).
- A scripting-API.
- Transform cells through command expressions.
- Stylus input optimization.
- Touch input optimization.
- Ability to control mouse/scroll sensitivity
- Ability to customize highlight color

- Ability to customize background pattern, image or color
- Ability to switch between light and dark modes
- Project manager to manage/switch between a collection of related sessions (this has to be considered carefully, it might be detrimental)
- Snapshot difference-visualization
- Audio-attachments
- Video-attachments
- Use a local camera to capture audio- and video-attachments
- Use a remote camera (e.g. on a smartphone) to capture audio- and video-attachments
- Drawings not constrained by cell bounds (automatically subdividing)
- More pen controls for drawing mode (3 sizes and color)
- Stickers and a sticker tool (for highlighting and commenting layouts and cards)

Collaboration features:

- Shared session server (sync state of shared session)
- Shared materials server (distribute stacks to a team)
- Project server (manage versions of sessions for a project and team)

Development features:

- Defined interfaces for all modules and add indirections based on them.
- Test automation.
- More robust behavior in fringe cases and deliberate attempts to break the application.
- Better error messages for field debugging.
- Metrics collection for test deployment.
- Evaluate an alternative, more secure technology stack.
- Consider an open source release.

Features relating to content:

- Glyph selector in edit mode (arrows, ordinals, box-drawing, etc.)
- Font switchable between variable-width and mono-width (some people like ascii-drawing)
- Text formatting shortcuts for bold, italic, underline, strikethrough, normal (comes up in interviews).
- Alt-drag to duplicate (important to pull cards from a utility stack).
- Pasting URIs to import materials
- Import materials from URIs (makes sharing of stacks easier and provides a maintenance mechanism)
- four categories of content: oblique influences (break free), reminders (stay on track), research materials (understand), personal notes (track progress), team notes (sharing).

Utility content (card stacks) that give users more ways to represent their ideas and offer additional sources of constructive friction:

- Curated card stacks for various contexts
- Cards representing design principles (proximity, common fate, contrast, negative space)
- Cards representing rhetorical concepts (question, answer, open end, A/B, pause/ponder, emphasis, repetition)
- Cards representing rhythmic patterns
- Cards with a few signaling colors
- Avoid: Symbols and Icons (too specific), extensive color palettes (not a design tool).

Bibliography

[Bas99] R. Baskerville. Investigating information systems with action research. *Communications of the Ais*, 1999.

[Ben09] B. Benderson. The promise of zoomable user interfaces. In *VINCI '10 Proceedings*, 2009.

[DiS15] C. DiSalvo. *Adversarial Design*. The MIT Press, 2015.

[ES75] Brian Eno and Peter Schmidt. Oblique strategies, 5th edition. Self-published, 1975.

[Fle01] Alan Fletcher. *The Art of Looking Sideways*. Phaidon Press Limited, London, 2001.

[Gam97] Erich Gamma. *Design Patterns: Elements of Reusable Object-oriented Software*. Addison-Wesley professional computing series. Addison-Wesley, Reading, Mass., 1997.

[Gre07] Chris Green. Improved alpha-tested magnification for vector textures and special effects. In *ACM SIGGRAPH 2007 Courses*, SIGGRAPH '07, pages 9–18, New York, NY, USA, 2007. ACM.

[Kah11] D. Kahneman. *Thinking, Fast and Slow*. New York: Farrar, Straus and Giroux, 2011.

[Kau93] Stuart A Kauffman. *The origins of order : self-organization and selection in evolution*. Oxford University Press, New York ; Oxford, 1993.

[Lam86] Leslie Lamport. *LATEX : a document preparation system*. Addison-Wesley, Reading, Mass., 1986.

[Lau93] B. Laurel. *Computers As Theatre, 2nd Edition*. Addison-Wesley Professional, 1993.

[LKD05] Sonja Lyubomirsky, Laura King, and Ed Diener. The benefits of frequent positive affect: Does happiness lead to success? *Psychological Bulletin*, 131(6):803–855, 2005.

[Lun01] Arnold Lund. Measuring usability with the use questionnaire. *Usability and User Experience Newsletter of the STC Usability SIG*, 8, 01 2001.

[Mar09] Robert C. Martin. Clean code, 2009. Dateiformat: HTML.

[McC04] Steve McConnell. *Code Complete*. Microsoft Press, Redmond, Wash., 2 edition, 2004.

[MSS+18] Maximilian Mackeprang, Gerold Schneider, Johann Strama, Philipp Kuhnz, , and Claudia Müller-Birn. Kaleidoscope: An rdf-based exploratory data analysis tool for ideation outcomes. In *The 31st Annual ACM Symposium on User Interface Software and Technology Adjunct Proceedings*, UIST '18 Adjunct, pages 75–77, New York, NY, USA, 2018. ACM.

[Nor13] Don Norman. *The Design of Everyday Things*. Basic Books, revised, expanded edition edition, 2013.

[ode10] *Oxford Dictionary of English*. Oxford University Press, Oxford, 2010.

[Ram06] W. Rammert. *Technik-handeln-wissen*, chapter Technik, Handeln und Sozialstruktur: Eine Einführung in die Soziologie der Technik. Springer VS, 2006.

[RM15] Paul Ralph and Rahul Mohanani. Is requirements engineering inherently counterproductive? In *Proceedings of the Fifth International Workshop on Twin Peaks of Requirements and Architecture*, TwinPeaks '15, pages 20–23, Piscataway, NJ, USA, 2015. IEEE Press.

[SCGD15] Pao Siangliulue, Joel Chan, Krzysztof Z. Gajos, and Steven P. Dow. Providing timely examples improves the quantity and quality of generated ideas. In *Proceedings of the 2015 ACM SIGCHI Conference on Creativity and Cognition*, C&C '15, pages 83–92, New York, NY, USA, 2015. ACM.

[SHP+11] Maung Sein, Ola Henfridsson, Sandeep Purao, Matti Rossi, and Rikard Lindgren. Action design research. *Management information systems : mis quarterly*, 35(1):37–56, 2011.

[Sim69] H. Simon. *The Sciences of the Artificial, 3rd Editon*. The MIT Press, 1969.

[Sto10] E. Stolterman. Concept driven interaction design research. *Human-computer Interaction*, 2010.

[SZ03] Katie Salen and Eric Zimmerman. *Rules of Play: Game Design Fundamentals*. MIT Press, 2003.

[Tal07] Nassim Nicholas Taleb. *The Black Swan: The Impact of the Highly Improbable*. Allen Lane, U.K., 2007.

[Tuf01] E. Tufte. *The Visual Display of Quantitative Information*. Graphics Press, 2001.

[Uni19] Unity Technologies. *Unity Manual*, 2019.1-002w edition, 2019.

Appendices

A Explorative Research Observation Notes

The following list describes the percepts[1] that could be identified through the data collected during the observed ideation workshop at CeRRI. The open coding underlying this summary is included in appendix C.

Linear browsing: In interviews it was repeatedly mentioned that going for a walk was helpful for ideation. During group discussion and in interviews it was observed that the sequential nature of dialogue seemed to transform the participants understanding of the problem space incrementally. It was mentioned that a ordered discussion is important so that individual ideas can be given time for consideration. Concurrent discussions were not favored. During the workshop is was observed that participants often processed stacks of cards in sequence, one at a time as part of a classification process or to prompt associations.

Spatial arranging of cards: Participants repeatedly expressed the wish to arrange materials (cards, examples, notes) on a table or wall. Various reasons were given and observed, among them classification, signifying change to others, as self signifiers and prompts for reflection form a particular angle (signified by the spatial interaction). Also participants often changed their position, walking around the space where materials were arranged, taking alternative position relative to the materials and in the room (the physical space where ideation takes place was mentioned as being influential).

Repeated reset: A pattern of participants repeatedly starting fresh was observed. On small and larger scales. Idea threads are abandoned and new ones started. Sometimes captured by note-taking. Resets were also somewhat represented in the workshop design, with each phase changing the mode of interaction and setting out to create a new landscape of understanding. Arrangements of cards are created and repeatedly abandoned. It may be supposed that arrangements become progressively less meaningful or useful as they grow, maybe because they become progressively worse at representing the actual relationships between items, overview gets lost and their processing becomes more tedious.

Leaving and resuming: Participants expressed a strong influence of time constraints on the ideation session and wished for self determined rhythm of the ideation, allowing for time to do further research, take breaks or talk to non-participating experts. In general the wish to document current states after phases was present, although its utility was questioned. In general the idea of leaving and resuming a session, preserving its state (in the shape of the arrangement of materials) was perceived as nice to have but impractical.

Legitimacy of methods and tools: In participant's reflection of known information systems supporting ideation (exemplifies by digital whiteboards) their inherent limitations were classified as so severe that even affordances not available

1 percept: A mental concept that is developed as a consequence of the process of perception. [ode10]

through manual/analog methods made them appear useless as general tools for ideation. It seems the whole genre of ideation support software is viewed with disdain. The direct question of how information systems would be able to support ideation yielded no pertinent response. It seems the utility of information systems is seen or understood only in the shape of hypertext browsing, search engines and document processors.

Information processing rate: Participants mentioned limits to how much information they able to absorb and retain in a given length of time, by what means it is provided, i.e. whether it was optimized to be absorbed quickly. Participants repeatedly mentioned the textual nature of the materials used during the workshop made it somewhat difficult for them to quickly process the available information and to get a good overview at a glance. They mentioned limits as to how much information they are able to absorb in a given length of time depending on the means by which it is provided, i.e. whether it was optimized to be absorbed quickly or whether the media type was suitable for the message. Some participants also stated that the textual nature was no problem for them.

Permission to break things: Participants repeatedly expressed a wish to be free to use all surfaces in their vicinity for ideation. In particular the walls and tables themselves, without any intermediate, for drawing and writing. This hints at them feeling constrained by neat and formal environments that must revert to a neutral state immediately after the end of a session. Effectively such an environment prevents an ideator from walking away from a session and resuming it. Some participants intimated that they would have liked to have some bold means of expressing themselves and seemingly found the small index cards and text markers used in the workshop to be rather tame. Participants displayed various affinities towards order and messiness, both in descriptions of their preferred process, environment and approaches to physically organize/arrange materials.

Gaining and maintaining overview: Participants expressed the initial desire to orient themselves relative to whatever context or materials they were given. I.e. all participants began their deliberation of the idea-sparks by wondering how the subset they were given related to the whole set. Once overview was achieved and could be used to identify opportunity areas the focus turned towards detailed examination if these areas. Overview was variously used to balance attention, for backtracking, finding alternative branches and discovering overlooked opportunity areas.

Embracing ambiguity (vagueness): The ideation process was variously described as starting as a big mess that gets sorted, discussed, shifted and turned until some thread is discovered that seems to lead somewhere.

Transformation of ideas into a persistent form: It was variously mentioned that the process of translating an idea from the mind to a shareable medium like drawing a picture of it, describing it verbally or writing it down is a critical component of effective ideation. It requires decisions to be made, thereby clarifying the idea in the mind, revealing problems with it and sometimes new opportunities. The

translation appears to serve as favored way to establish a legitimate inner dialogue. As a result the idea becomes shareable and prompts additional dialogue with others. Some participants characterized themselves as visual thinkers and mentioned the lack of opportunity to express their ideas by drawing as limiting.

Sharing and passing ideas: It was variously mentioned that the presence of other ideators, their materials and interactions with them (gesture and dialogue), including non-constructive feedback influenced them significantly in their perceived progress towards discovering a good idea.

Diversity of materials and other inputs: Participants mentioned variously that absurd, strange, silly or misconceived sparks were quite memorable and also lead to some of the more interesting reversals, branches or leaps in their thinking. Contemplation of diverse materials supposedly produces better (more creative) ideation outcomes.

Familiarization: Participants described the initial phases of an ideation process variously as a time of getting to know the problem, the influences acting on it. In case of the technology that was subject of the observed ideation, there was much interest in the precise capabilities of the technology. It seemed like participants were exploring the boundaries of the technology, exploring possible edge cases rather than looking for opportunities at the heart of the technology, i.e. any obvious applications. This would indicate that opportunity areas for possible innovations are seen where the technology reaches the limits of its capability (edge of chaos, adjacency).

Convergent and divergent thinking: Participants described their thinking as repeatedly alternating between divergent and convergent modes. Likely inspired by the "Double Diamond"-model of a design process and other "Design Thinking" concepts. Some participants considered the temporal separation of these modes as significant drivers of group productivity, primarily to maintain focus and cohesion of the group's shared conceptual state.

Trial, error & prototyping: Participants expressed a desire to try out ideas and play around with them. An example was given where fast prototyping was used to test ideas, paper prototyping was mentioned. Participants intimated that prototyping is usually an integral part of a design or ideation process. It is likely that the nature of the observed workshop, in particular the limited access to materials and time constraints prevented such testing and experimenting. Consequently it was not observed. Since it also wasn't described as critical, it seems the value attributed to it is speculative. Certainly it provides complementary value, it is however likely that it does not help much at finding new ideas. This would also be explainable by the convergent character the building of something has, it is decidedly better at identifying problems (like transformation) than at exploring opportunities.

Taking a walk: Participants mentioned that they routinely had experiences of having good ideas while going for a walk or generally moving around in a physical space. Supposedly while being at ease, without pressure or other things than the

ideation subject occupying their mind. Possibly related to this notion it was found experimentally by [SCGD15] that typed and timed examples of ideas can significantly aid creativity if they are received on demand.

Forming ideas from materials: Participants mentioned variously that they like to involve physical materials into the ideation process. These include models of the problem space, gadgets, artifacts from the problem space, artifacts they use for inspiration (examples, tokens, representative objects, descriptions, pictures, toys and other things). It appeared, the physical nature was seen as important maybe introducing a tactile sensation, that provides a helpful, decidedly non-cognitive and rather more emotional stimulus to some. It is unclear how effective the involved physical component is towards creating a cognitive effect that leads to good ideas. It is supposed that many of the materials effective qualities can be replicated in a virtual form.

Connecting ideas to reality (with a plan): It was mentioned that ideas and innovation are distinctly separate concepts and that their difference is actually very important for understanding the dynamics of a particular ideation process. The observed workshop concerned itself with innovation, which was described as having an actual effect on the world or a concrete plan on how to implement it, where an idea is a weaker concept that encompasses basically everything.

Playing the game (challenges, resistance, obstacles): The idea of antagonism, obstacles and challenge as a constructive feature (as exploited in games) was not directly mentioned by participants. However the observation of the workshop gave a strong impression of playful interaction and even setup: introducing difficulties, challenges and contexts only as triggers for creativity. It was mentioned that any state during the ideation process has not value in itself, it is merely a means to transition to the next state, ultimately ending when the participants leaving the session with ideas in mind that they then, in a separate process transform into a persistently valuable representation.

Transitioning between ephemeral states of mind: Participants intimated that the purpose of processes like clustering in the context of ideation primarily serve as catalysts for the transformation the mental state in its participants. What can be perceived about the process visibly is actually not a good representation of what is really going on. Its can only serve as trace evidence. Actually at the end of any segment during the observed workshop, it seemed that the participants carried away a decidedly personal mental model of the problem situation and potential solutions that were not captured by any visible artifact.

B Prototype implementation

This chapter provides additional details of the Stacktree pattern implementation Periscope 0.5. It builds directly on section 4.5 and continues with a discussion of technology selection and a description of the Unity development platform and API that was used for constructing the Periscope 0.5 application. Following that is a description of Periscope 0.5's key components, the tree, card and stack, a few organizational notes and finally a very brief tabular overview of all the classes that compose the system.

B.1 Technology selection

The fundamental paradigm in the design of Periscope 0.5 is one of creating behavioral entities that will be configured by the user to create complex implicit interactions and effects through the user's eyes. Realizing this kind of interaction means, that a large part of the software would simulate an environment or world in which objects can be placed and manipulated. This world would have to have a variable viewport which would display objects of various sizes. One approach would be to re-size the objects to fit on demand. This would however involve many unnecessary calculations just for zooming in on an object while actually none of its data would change as a result of this interaction. So the choice was made to consider all objects fixed in space unless their relative position to each other is supposed to change. The natural approach for modeling such a relationship would be one of a camera looking into a space. And this immediately means that traditional GUI frameworks would have a hard time adapting to it as they generally offer no concept of a camera, projection or zooming. In fact simple animated transition are beyond their design's scope. Naturally more recent developments have added such functionality, for example into Qt[2]. However the simplest approach — and one that also satisfies a number of other requirements growing from the conceptual design quite naturally — lies in looking at game engines to serve as framework for developing a Periscope 0.5 prototype. The advantage of this approach is that a game engine is designed for creating the widest possible range of experiences, making as few assumptions as possible about how a user will interact with the finished application. This means, all patterns and architecture styles are rather abstract and generic, geared towards iteration, interchangeability, and a separation of scene construction (by designers) vs. behavior definition (by engineers).

B.2 Software, libraries, frameworks

This section describes the tools and readymade components actually used in the construction of the Periscope 0.5 prototype with a brief note on the reason for

2 Qt is the faster, smarter way to create innovative devices, modern UIs & applications for multiple screens. Cross-platform software development at its best. Quoted from metadata at `https://www.qt.io/`, retrieved Monday 1st July, 2019

which they were chosen.

Development platform: Unity is a real-time development platform for hardware accelerated 3D, 2D, XR visualization. It uses mono (an open source implementation of .NET[3]) with C# for scripting and accessing the engine API[4]. The Unity Engine is designed to be used in conjunction with an editor application that handles the design and construction of scenes (much like a GUI designer) by composing of objects from serializable classes and data driven dependency injection. This allows workflows in Unity to be split into design (arranging and configuring objects, handling assets, previewing design, testing) and engineering (coding, testing, version control, collaboration), each within their optimal environment. The choice was made based on familiarity and previous experience with it. Unity has the following expected limitations:

- A limited copy buffer (text only)
- No provision for receiving drop events from the OS
- Not designed to be power-conserving (it is frame rate driven instead of event driven)
- Provided UI widgets are very basic

And provides the following opportunities relevant to this project:

- No GUI dogma
- A camera
- Designed around hardware acceleration
- Much control over rendering
- A mature editor for data-driven GUI and scene design
- Provided UI widgets are suitable for prototype mockup and easily extendable.
- Mature library support via .NET
- Cross platform build targets (Mac, Win and WebGL).

Font Rendering: Periscope 0.5 is designed to have a zoomable UI which is also adaptive to various screen resolutions. This means the same text components may have to be rendered at hugely different sizes (approx. 3 orders of magnitude). The basic Unity font renderer relies on scaling pre-rendered font-textures causing text to display nicely only in a narrow range of sizes relative to the original font texture. To make text more pleasant to read an alternative text renderer, TextMeshPro (TMP) is used. TMP uses a signed distance field for each character, encoded into the pre-rendered font-textures[5]. Through this additional information crisp scaling

3 .NET is a developer platform with tools and libraries for building any type of app, including web, mobile, desktop, gaming, internet of things (IoT), cloud, and microservices. *Quoted from the meta description of* `https://dotnet.microsoft.com`*, retrieved on Monday 1st July, 2019*

4 Application programming interface API is a set of clearly defined methods, protocols and tools for programming a software. an API is provided by a system that provides building blocks for a programmer to construct their software.

5 This method of efficiently scaling character glyphs was described in detail by C. Green at SIGGRAPH'07 [Gre07].

of shapes over a larger range is possible a roughly the same cost as mipmapped textures and bilinear scaling. The library choice was made on a trusted-first-fit Trusted First Fit (TFF)[6] basis.

System File Dialog: Periscope 0.5 can import text- and image-files that can be used to create cards in a batch operation or to attach individual ones to particular cards. This provides the opportunity to construct and quickly load various test scenarios during evaluation. Since Unity provides no file browser the open source wrapper for native file dialogs by Gökhan Gökçe was used[7]

Linear Assignment Problem: Periscope 0.5's quadtree layout arranges a region's arbitrarily distributed bucket items in an orthogonal grid by minimizing the aggregate positional error in the region. The involved linear assignment problem is solved by the a Kuhn–Munkres implementation taken from the accord.NET[8] machine learning framework. The details of the implementation weren't investigated as even the time complexity in $O(n^4)$ of the original algorithm is acceptable for arranging a maximum of 16 tiles per update and an improvement beyond $O(n^3)$ has not yet been found. Chosen via TFF. For the evaluation build of Periscope 0.5, the tree's bucket size was configured at 1, so the algorithm was actually bypassed by a strategy pattern.

Programmatic Animations: Periscope 0.5's UI extensively uses animations to visually communicate the state transitions of entities. C#'s coroutines (`IEnumerator` and `yield return`) that are the default model of concurrency in Unity offer a clean way to programmatically animate Unity-object's member values via extension methods. Among various options, the DOTween library[9] was chosen to handle programmatic animations in this project. Chosen via TFF.

Asset Creation: For the prototype development and asset creation the following applications were used: Unity Editor (Unity scene construction, build), JetBrains Rider (development), Adobe Photoshop (raster graphics), Adobe Illustrator (vector graphics), VisualStudio Code (text editing). The choices were made based on familiarity and previous work-experience with them.

6 Trusted-First-Fit TFF: the first option minimally satisfying requirements and recommended by two independent, trustworthy sources is chosen.

7 Unity Standalone File Browser by Gökhan Gökçe — A simple wrapper for native file dialogs on Windows/Mac/Linux. `https://github.com/gkngkc/UnityStandaloneFileBrowser`

8 The Accord.NET Framework is a .NET machine learning framework combined with audio and image processing libraries completely written in C#. *Quoted from* `http://accord-framework.net/`, *retrieved Monday 1st July, 2019*

9 DOTween (by Daniele Giardini (et. al.)) is a fast, efficient, fully type-safe object-oriented animation engine for Unity, optimized for C# users, free and open-source, with tons of advanced features. *Quoted from* `http://dotween.demigiant.com/index.php`, *retrieved Monday 1st July, 2019*

B.3 Unity API

While Unity uses a largely standard instance of the mono[10] .NET implementation for accessing the engine API, it also employs a number of concepts and procedures that are specific to Unity which must be explained to understand the way Periscope 0.5 (or any other project implemented with Unity) is constructed.

B.4 Unity Core

One approach to understanding the Unity API is looking at it as a general purpose visualization platform or glorified GUI framework with a strong emphasis on performance and prominent focus on being agnostic towards what projects are realized with it. At the core of the Unity API is a composition system consisting of `GameObject`-entities with attached Behaviors. Behaviors are serialized classes deriving from `Component`, All Instances of a `GameObjects` are dependent upon a `Scene` which locates them in a three dimensional space and updates them in a loop as fast as possible. Components of `GameObject` receive lifecycle event calls (see figure 2) from the engine if they derive from `Component` → `Behavior` → `MonoBehavior`. This way Unity takes full control over the lifecycle of these objects and over their lifetime they receive a sequence of methods calls by the Engine. The serialization of all components and scenes enables construction and instantiation of scenes in a data driven way.

Generally this data is edited by a GUI application (the Unity Editor) or custom scripts and not editable manually. The Unity Editor itself has a GUI that is easily extensible directly as part of any project, enabling custom object-inspectors or editors and tools as needed with little overhead. Unity puts hardly any constraints on the use of external libraries. A project can easily separate its data model and control from the engine and only use it as presentation layer. However a powerful feature of its data driven architecture is the the intuitive and well organized dependency injection and entity hierarchy that can be constructed through the editor. Figure 1 illustrates the relationships of Unity's most important classes.

Each Unity scene defines a world space for its contained `GameObjects`. Each `GameObject` has a dedicated `Transform`-component that represents its position, rotation and scale in the scene and derived from it a local space. For `GameObjects` representing 2D entities, a `Transform`-derived `RectTransform`-component is used to additionally represent the anchors and pivot of a rectangle on a specific `CanvasPlane` in the scene. This `RectTransform` component is used in all visible components of Periscope 0.5. Entities that are descendants of a `Canvas` Entity have access to a `EventSystem` designed for constructing graphical user interfaces. The associations between `Canvas`, `EventSystem`, `CanvasRenderer` and `LayoutElement` are illustrated in figure 3.

10 Sponsored by Microsoft, Mono is an open source implementation of Microsoft's .NET based on the Ecma standards for C# and the Common Language Runtime. *Quote from* `https://mono-project.com/`, *retrieved on Monday 1st July, 2019*

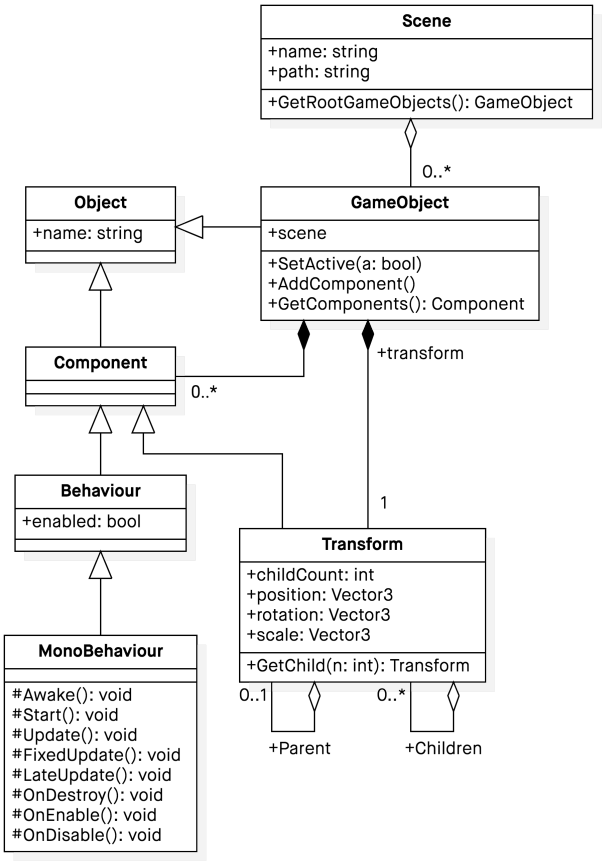

Figure 1: *A simplified view of the Unity engine core classes, as they are helpful to understanding the Periscope 0.5 architecture [Uni19]*

The visual representation of entities in Unity is facilitated by `Camera` components and various `Renderer`-derived components that interface with the Engine's render-loop and make objects appear visually in the world at their Transform's position. The Render loop in Unity is transparent to the user and I have found no information on the details of its inner workings. Periscope 0.5 for the most part uses the provided derivatives of the `GraphicRenderer`-component that are optimized for `Canvas` components. Like all `Renderers` in Unity it maintains a pool of shared textures, shaders and mesh data in GPU memory, making hardware supported transformations to the GUI very fast. For the most part, only transformation matrices have to be pushed to the GPU on each frame and redraws of the UI can be compartmentalized to avoid bottlenecks from suddenly pushing many large new textures to the GPU. Where usually in Unity the rendering of an object involves multiple components attached to a `GameObject`, the `GraphicRenderer` simplifies this structure into a single component that encapsulates, a polygon mesh, a filter that optimizes shared mesh data, a renderer and texture UV-mapped[11] to the

11 A UV-map is used in 3D-modeling to project a texture onto a mesh. At its core it is a

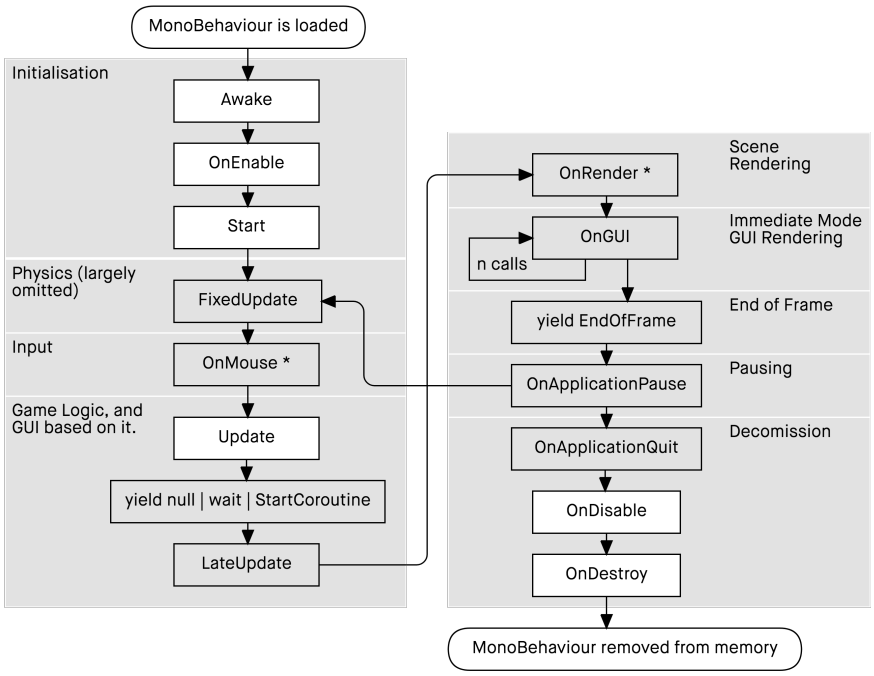

Figure 2: *A simplified flow diagram of the lifecycle of a Unity object. Physics, rendering details and internal the animation system are omitted [Uni19]*

mesh.

B.5 Unity UI

The performance oriented architecture of the Unity UI system somewhat constrains the things that can be done, out of the box, with it. Programmatically drawing textures for UI components from vector shapes or otherwise is not part of the design. The sprite- and texture-based GUI system, which is effectively just a purposeful extension of the basic Unity render model builds its widgets using the same concepts as a performance optimized 3D scene would, by once pushing the full object state to the GPU and subsequently only using transformations on vectors like position, rotation, scale, blending and functional programs (shaders) to modify the visual appearance. This performance-aspect and custom design of widgets are usually of no interest to the designer or developer of flat, standard, 2D interfaces. Most widely used GUI frameworks make little use of hardware acceleration and offer little to make widget design less technical, usually for good reason (standardization improves usability). Periscope 0.5's live zoom-functionality, various animations and generally the desired fluidity of interaction, however require

table that maps each vertex of a mesh to a specific coordinate (u, v) in a 2D texture. During rendering any defined triangle between a given set of vertices is (among other things) a projection of the texture that lies between their associated uv coordinates.

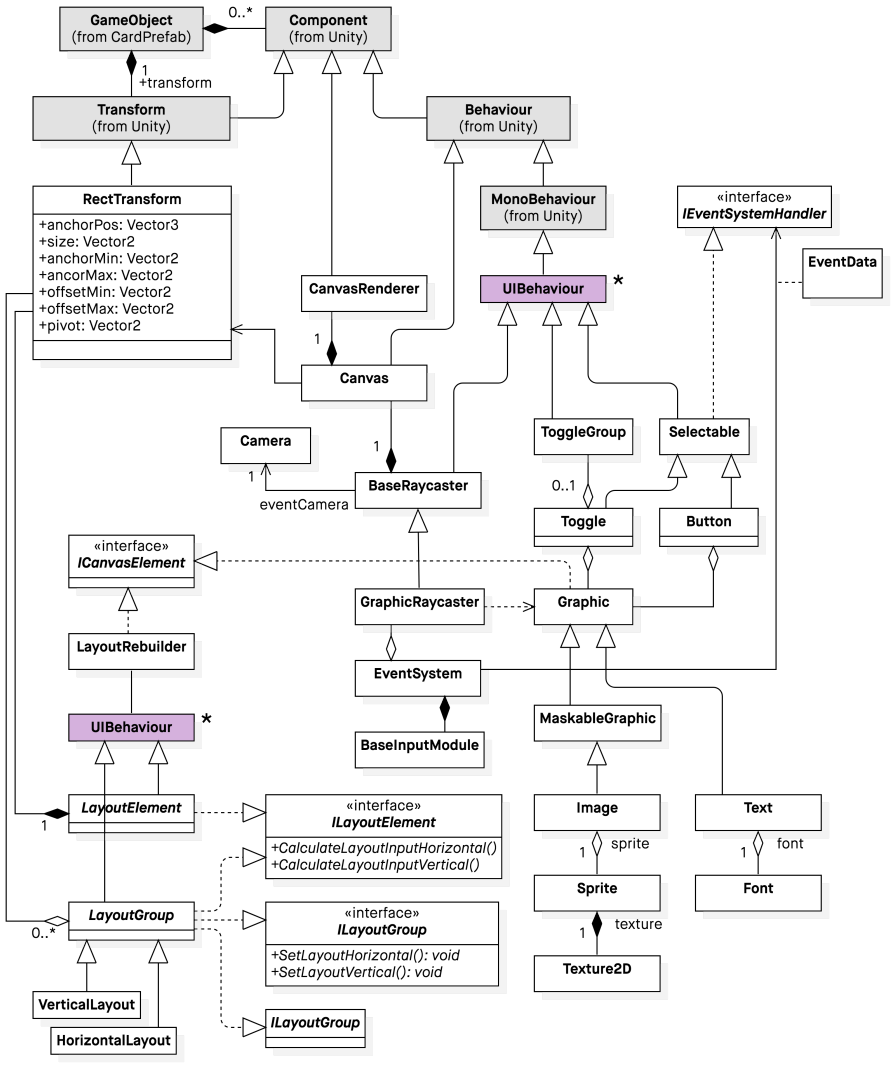

Figure 3: *A simplified view of the* `UnityEngine.UI` *classes most relevant to the Periscope 0.5 architecture, illustrating the compositional nature of the* UI *framework, based on* `GameObjects` *and showing interface-hooks into the layout- and event-system which are extended by Periscope 0.5. [Uni19]*

a design that is decidedly conscious of these constraints. To illustrate, figure 5 illustrates a decomposition of the Card widget into its various `Graphic`-layers as a consequence of this design constraint.

B.6 Organizational concerns

For Periscope 0.5 the Unity scene-hierarchy was used for maintaining a clear structure that groups entities semantically and decoupling them where ever possible. It was also used in places as data structure for layout groups in line with the overall architecture of the underlying UI system. Naturally it also enabled a data driven, declarative composition of scenes and traceable dependency injection. Since all members of Components attached to GameObjects can be inspected in the Editor, this offered a comfortable and efficient environment for experimenting with various configurations and tuning the final product.

Since Periscope 0.5 is a high function prototype, best practices [McC04], [Mar09], [Gam97] have been employed when possible. However as iterative changes happened, the cleanness of the system's module coherence and decoupling was demoted in priority behind productivity. After Periscope 0.5 was feature complete a mild refactoring was undertaken to correct the most egregious offenders to good form, however in order to not introduce unnecessary bugs before evaluation, larger changes fundamentally altering the class- or scene-structure were avoided.

The project's scope and prototypical nature did, not yet warrant the writing of unit tests as the algorithms were easy enough to validate via assertions and manually triggering edge cases. The test automation of GUI interaction in Unity isn't well supported and was done manually to stay within the project's scope. Also with only one developer, bugs were still rather easy to trace and resolve. Further iterations would however need automations as the manual test routine is getting ever more unreliable.

B.7 Tree module

The tree map at the center of Periscope 0.5's UI is realized as a custom layout component utilizing update triggers from Unity's built-in layout system. The basic idea is this: Cards are at their root ordinary `RectTransform` objects that can store position and size, which is sufficient to describe a square in a 2D-plane. Indexed by their 2D position these can be inserted into a quadtree data structure which will subdivide based on the proximity of the added `RectTransforms`. The quadtree's subdivision pattern is then used to set the sizes of all associated `RectTransforms` so that they cover the quadtree region they are contained in. The object hierarchy constituting a card then responds to this change in size by cascading update calls through all descendent layout components. The modification of the cards' arrangement by the user can be captured by a drag & drop behavior.

Since such a drag event can be temporally abstracted into only occurring when the drag-pointer leaves the visual boundary of a card, a redraw of the UI happens only then and only has local effects on at most four cards (its previous and current neighbors). This provides significant opportunity for strategic optimization while still giving the user live feedback on their interaction.

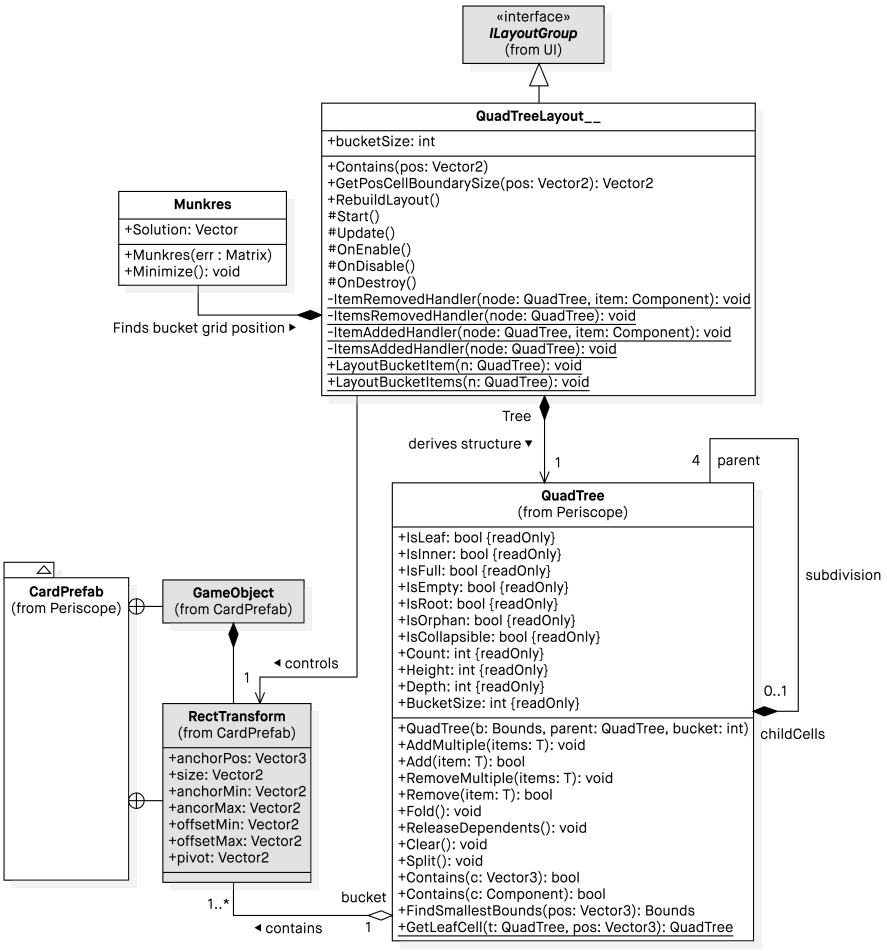

Figure 4: *The physical architecture of the quadtree layout component: A cell's root transform is updated by the* `QuadTreeLayout` *when a layout rebuild is triggered. This triggers updates to all layout components descendent of that same root.*

`QuadTreeLayout` maintains a quadtree data structure containing all positions of child nodes below a given root `RectTransform`-object. When the Unity layout system signals a layout update event, `QuadTreeLayout` clears the quadtree, collects all child `RectTransform`-objects of its target `RectTransform`-object, and adds them into the tree one by one, indexed through their world space position.

items ← set of RectTransform objects to layout
Q ← QuadTree for RectTransform objects, bucket size n, $\sqrt{n} \in \mathbb{Z}$
On SIGNAL → REBUILD(Q, *items*)

```
procedure REBUILD(Q : QuadTree, T : RectTransform[])
    Clear Q
    for item ∈ T do
        Add item to Q
    LAYOUT(Q)
```

This tree Q then contains a structured set of regions, implicitly sized based on the depth in the tree a given position ends up in. Each subdivision of a region creates four equally sized new regions. In a second pass through the tree, each `RectTransform` item contained in a region is assigned dimensions matching those of its containing region. Depending on the bucket size n of the tree, a region may be shared by multiple items. In this case the region is divided into n equal tiles arranged in a orthogonal grid and the contained item's are scaled and moved into this grid.

```
procedure LAYOUT(Q : QuadTree)
    if Q is INNER then
        for child ∈ Q.children do LAYOUT(child)
    if Q is EMPTY then return
    if Q is LEAF then
        r ← √n
        w ← Q.width/r
        h ← Q.height/r
        tileSize ← (w, h)
        tiles ← (position, size)[n]
        for y ← 0, r do
            for x ← 0, r do
                tileCorner ← (x · w, y · h)
                tileLocalPos ← tileCorner + tileSize/2 − Q.size/2
                tileWorldPos ← tileLocalPos + Q.position
                i ← x + x · y
                tiles[i].position ← tileWorldPos
                tiles[i].size ← tileSize
```

The decision which item is placed into which grid tile is a linear assignment problem that can be solved in $O(n^4)$ which is acceptable for the current bucket size of four. Significant optimization would be possible by making the layout rebuild differential and only solving the assignment problem for the regions that have changed from the previous state.

```
        E[n][n] ← n × n error Matrix
        for i ← 0, n do
            for t ← 0, n do
                E[i][t] ← || tiles[t].position − Q.bucket[i].position ||
        S[n] ← solve LAP for E using Kuhn–Munkres
        for i ← 0, n do
            Q.bucket[i].size ← tileSize
```

$$Q.bucket[i].position \leftarrow tiles[S[i]]$$

The quadtree data structure used for the layout procedure above is a custom implementation of a point-region quadtree. Each region in the quadtree can contain a number of points. Each point is associated with a data object derived from `Transform`. The `Transform`'s world position is used as index for the insertion into the tree. Since `Transform` derives from `Component` and is attached to a `GameObject`, each `Transform` in the quadtree also provides an access point to the Unity scene hierarchy starting from a particular `GameObject`. In this implementation each tree node is itself represented as a tree. Each node, including the root node, is bounded and may only contain points that lie within it (this allows multiple trees to exist side by side).

```
isInner(Q) → ∀c ∈ Q.children : c ≠ null
isLeaf(Q) → ∀c ∈ Q.children : c = null
isFull(Q) → |Q.bucket| = n
procedure Init(pos, size, parent, n)
    Q = (center, size, parent, children, bucket, n)
    Q.children[4] ← 4 child regions
    Q.bucket[n] ← set of size n
    Q.size ← size
    Q.center ← center
    Q.parent ← parent
    Q.n ← n
    return Q
```

To ensure that unused tree structures get cleaned up by the garbage collector, and to reduce opportunity for undefined behavior caused by orphaned references, the tree's CLEAR procedure explicitly breaks all links to other objects.

```
procedure Clear(Q : QuadTree)
    for c ∈ Q.children do
        if c is leaf then
            Q.bucket[0 ... n] ← null
            Q.parent ← null
        if c is inner then
            Clear(c)
        Q.children[0 ... 3] ← null
```

An object is added to the tree via its position. The procedure checks if the given node bounds contain the position, if not it returns FALSE, if it does it proceeds to check if it is a leaf node. If it is a LEAF and also not full, the object is added, if it is full, the node is split, turning it into an inner node. If the procedure encounters an INNER node it recursively calls ADD on all child nodes for the same object, which causes the three out of bounds calls to fail, and succeeding on the child node that contains the position.

```
procedure Add(Q : QuadTree, t : Transform)
```

```
if ¬(Q contains t) then
    return FALSE
if isLeaf(Q) then
    if isFull(Q) then
        Split(Q)
    else
        Q.bucket ← Q.bucket ∪ t
        return TRUE
if isInner(Q) then
    for c ∈ Q.children do
        success ← Add(Q, t)
        if success then
            return TRUE
return FALSE
```

A call to Split redistributes the objects contained in a region to newly created child region based on their position. The algorithm doing this (below) favors the north-west quadrant. I.e. a newly added position in most cases ends up in either of the other quadrants. This affects the UX by making the north-west seem more important due to its stronger resistance to change.

```
procedure Split(Q : QuadTree)
    size ← Q.size/2
    for i ← 0, n do
        quad ← (i/2, i) mod 2
        center ← Q.center + (quad − 1/2) · size
        children[i] ← Init(center, size, Q, Q.n)
    for item ∈ Q.bucket do
        for child ∈ Q.children do
        Add(child, item)                ▷ will succeed only for one child
```

B.8 Card Module

Cards are easily the most complex entities in Periscope 0.5 and the primary element the user interacts with. They have to facilitate, dispatch or route most of the intentions a user wants to realize. Additionally they have to clearly communicate their current state and its transitions.

The implementation of all the requirements imposed on them led to a set of classes that are quite friendly[12] with each other, in particular through certain promises made by the Unity Engine and application architecture which serve in

12 Friendly classes in this context share certain assumptions about each other's behavior, surrounding component structure and coherence with certain construction principles like dependency injection and inversion of control. These are fairly explicit if the architecture and design of the system are understood, but rather obscure when viewed only from a code perspective.

places as an implicit interface specification[13]. Where possible dependency injection and inversion of control is used, like in the rest of Periscope 0.5, to decouple component classes from their runtime context. Expectations placed on dependencies mostly stem from their behavior specified by the Stacktree pattern. It should be fairly easy to insert a level of indirection into the current implementation and establish a well defined set of interfaces between modules.

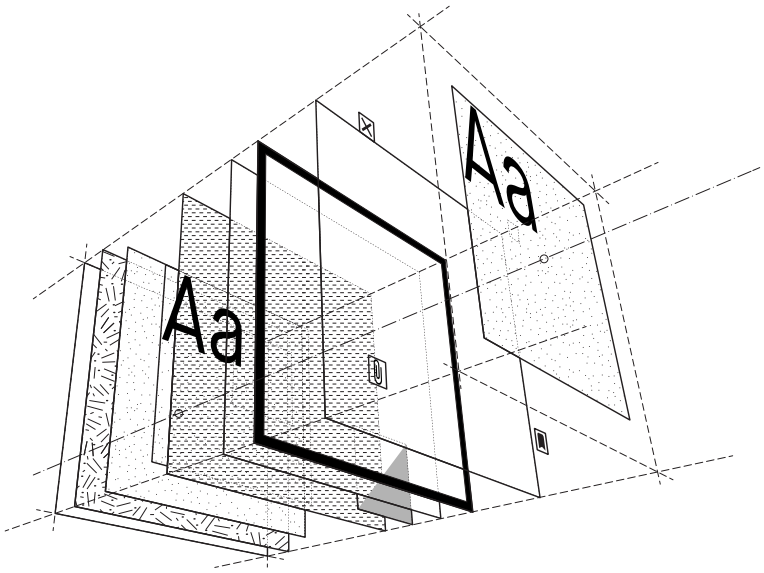

Figure 5: *A visual decomposition of the custom card widget. Illustrating the various layers involved in presenting its state; plus various command triggers.*

As mentioned above in section B.3, the Unity API benefits from a certain approach to construction of interface widgets. In case of the card widget it is best to imagine a card as a stack of layers[14] (figure 5), where each layer has a specific responsibility with visual or functional component. Components that respond to events are called *triggers* in Unity. They implement a specific Interface for each event they intercept. Events that are dispatched by the event system occur once at a specified time in the update cycle (figure 2). For this project the concept of a trigger was expanded to include any component that intercepts user actions, interprets them and dispatches actions that realize their intent. A formal command pattern is not implemented in Periscope 0.5.

B.9 Stack module

Each card can be dropped in either a tree map region or any of the stacks on the side of the screen. The root objects of each of these have a drop receiver component that places the received cards in their local data structure and keeps track of

13 Example: Each `GameObject` has a `Transform` component.
14 These are visual layers, not architectural layers

Table 1: *The layers that compose the card widget*

	Text editor	Provides text editing behavior to update the card text.
	Drawing canvas	Handles draw signals and displays their result.
	Split graphic	Signifies card split and intercepts activation events
	Duplicate graphic	Signifies card duplication and intercepts activation events
	Split & Duplicate trigger	Dispatches card split and duplication signals.
	Attachment graphic	Signifies attachment and intercepts activation events
	Attachment trigger	Dispatches attachment signals.
	Bookmark graphic	Signifies deletion and intercepts activation events
	Bookmark trigger	Controls the bookmarked state.
	Delete graphic	Signifies deletion and intercepts activation events
	Delete trigger	Dispatches deletion signals.
	Drag graphic	Signifies a drag event.
	Bookmark graphic	Signifies the card having a bookmarked tag.
	Selection graphic	Signifies the card's selected state.
	Icon graphic	Displays an icon signifying a card.
	Text graphic	Container for card text.
	Text background graphic	Makes text legible when overlaid on a picture
	Picture attachment graphic	Container for a picture attachment.
	Video attachment graphic	Container for a video attachment.
	Hotkey trigger	Responds to keyboard commands and dispatches corresponding signals.
	Select trigger	Makes cards selectable individually and as a group.
	Text edit trigger	Handles activation of the text editor and receives its callbacks.
	Drag trigger	Makes the card draggable.
	State presenter	Controls the visual display of card state.
	Interactable controller	Controls the availability of signifiers and event interception for triggers based on relative size of the card.
	Card data	Stores card state
	Background graphic	Intercepts all events not handled above.
	`GameObject` & layout interface	Locates the card in the scene and responds to layout changes, propagating them to the layers above.

Module boundary Data Trigger Graphic & box collider Edit behavior

session variables representing this association. Stacks and the trees exist each in a different vector space, the Stacks are an overlay in the screen coordinate space while the tree lives in a separate world space that can be projected through a camera to enable zooming and panning. Moving cards therefore also requires a transformation of their visual representation between these spaces which is handled by the respective drop receivers.

Each stack contains a behavior component which responds to input from the from a scroll-axis trigger. Unity aggregates mouse scroll-wheel input and scroll gestures from a touch-pad into this axis trigger. The cards currently associated with a stack are conceptualized as members of a double ended queue. In response to a scroll event, the head and tail of this queue are popped and then pushed into the queue at the opposing end. The actual data structure doing that stores the membership association between stacks and cards is transparent in the implementation of the Unity `Transform` component that was used for tracking this relationship. Using `Transform` for this purpose makes sense as the z-sorting of objects in a scene during rendering is based on their position in the hierarchy and sibling index. Therefore as an actual response to a scroll event, the sibling index of the first or last card on the container `Transform` is simply changed to correspond to the index at the opposite end.

To make this change in the sibling index visually traceable, a set of animations had to be sequenced between the actual index changes, with animation targets also depending on the size of the stack. Additionally animations needed to be interruptible by subsequent scroll events. Since this behavior is closely related to the details of the Unity and animation API an excerpt of the actual implementation illustrates the sequencing best.

```
private Transform _container
private Transform _previous
private const float DURATION;
public class StackFlow : MonoBehaviour, IScrollHandler {
    public void OnScroll(PointerEventData eventData)
        // [...]
        if (eventData.scrollDelta.y > 0) {
            var firstSibling = _container.GetChild(0);
            if (_previous == firstSibling) {
                firstSibling.DOComplete(true); // complete any running animation
                firstSibling = _container.GetChild(1);
            }

            void SlideIn() {
                firstSibling.SetAsLastSibling();
                firstSibling.DOMove(Rect.position, DURATION);
                if (_previous == firstSibling)
                    _previous = null;
            }

            firstSibling
                .DOLocalMove(Vector3.right * Rect.sizeDelta.x, DURATION)
                .OnComplete(SlideIn); // complete any running animation
                                      // then call SlideIn()
```

```
25
26              _previous = firstSibling;
27          } else if (eventData.scrollDelta.y < 0) {
28              var lastSibling = container.GetChild(container.childCount - 1);
29              if (_previous == lastSibling) {
30                  lastSibling.DOComplete(true);
31                  lastSibling = container.GetChild(container.childCount - 2);
32              }
33
34              void SlideIn() {
35                  lastSibling.SetAsFirstSibling();
36                  lastSibling.DOMove(Rect.position, DURATION);
37                  if (_previous == lastSibling)
38                      _previous = null;
39              }
40
41              lastSibling
42                  .DOLocalMove(Vector3.right * Rect.sizeDelta.x, DURATION)
43                  .OnComplete(SlideIn);
44
45              _previous = lastSibling;
46          }
47      }
48      // [...]
49  }
```

B.10 Application singleton

Control of the application itself, facilitation of modes and sessions management is handled by a monolithic singleton that contains all these. Description of its detailed functionality is deferred to the source code as its functionality is peripheral to the Stacktree implementation.

Sessions are serialized into JavaScript Object Notation (JSON), storing the state of all cards in value arrays, together with meta data supporting validation and conversion in case of changes to the serialization format. Below is an example of a serialized session containing two cards. The array values that share the same index represent a card.

Listing 1: *session.json*

```
1  {
2      ”Product”: ”Periscope”,
3      ”Version”: ”0.5.7”,
4      ”Build”: ”cef4f577e229d4af1a6ccdacbd0594fb”,
5      ”Company”: ”Public Void”,
6      ”Identifier”: ”com.publicvoid.periscope”,
7      ”DataPath”: ”/Applications/Periscope 0.5.7/Periscope 0.5.7.app/Contents”,
8      ”PersistentDataPath”: ”~/Library/Application Support/Public Void/Periscope”,
9      ”Timestamp”: ”06/15/2019 14:04:25”,
10     ”CollectionPath” : ””,
11     ”Container”: [1, 3],
12     ”Path”: [””, ”hello.jpg”],
```

```
13      "Text": ["Hello, world!", ""],
14      "X": [512.0, 512.0],
15      "Y": [512.0, -512.0],
16      "Tags": ["Bookmarked", ""]
17  }
```

When a session is saved all files referenced in it are copied into a collection-directory next to the session file. Additionally a separate recovery session file is created which is updated each time a change is made to the corresponding active session. This recovery file can be used to recover unsaved changes that were lost during a crash or unexpected termination of the application.

B.11 Other modules

Next to the behaviors implemented supporting the Stacktree pattern directly, a number of secondary behaviors had to be created. For the purpose of brevity they are only mentioned here by name and can be viewed in detail in the project repository (section 4.5.5).

Table 2: *All classes*

Class name	**Description**
AppTitleBox.cs	Displays the application title and build information.
Cell.cs	Stores the persistent state of a card.
CellAttachmentTrigger.cs	Allows file attachments to be added to cards.
CellBookmarkTrigger.cs	Allows card bookmarking.
CellDeleteTrigger.cs	Dispatches deletion of cards.
CellDetail.cs	A detail view of a region with various controls to adjust its state (currently unused).
CellDivideTrigger.cs	Dispatches division and duplication of regions.
CellDragHandler.cs	Allows cards to be dragged around.
CellEditHandler.cs	Allows cards to be edited.
CellHotkeyHandler.cs	Allows cards to respond to various hotkeys.
CellPresenter.cs	Manages the visual display of a card.
CellSelectHandler.cs	Manages the selection and marked states of a card.
CollapseGroup.cs	Generic behavior that hides a CanvasGroup based on a trigger.
ContainerId.cs	Enum storing container identifiers.
ContextMenu.cs	not implemented.
ContrastWithBackground.cs	Manages text color to have maximum contrast with its background.
CreateCellHandler.cs	Allows the creation of new cards.
DragGizmo.cs	Represents the gizmo that accompanies drag events.
DrawingCanvas.cs	Allows cards to be drawn on.
FlowLayoutGroup.cs	An adaptive grid layout.
ICellDrop.cs	Interface for all components that function as drop receivers.
InputRouter.cs	Translates raw input into application specific commands.

Class name	Description
InteractableOnSize.cs	Sets the interactable state of interface controls based on their relative size.
LevelOfDetail.cs	Manages the visual presentation of cards based on their relative size.
MarkAllCellsUnderTarget.cs	Allows the marking of all cards below a given node in the hierarchy..
MessagePresenter.cs	Displays a global log message.
ModalCanvas.cs	Shows a modal dialog.
ModalTrigger.cs	Captures and blocks all unhandled pointer events.
OutOfBoundsHandler.cs	Handles all events that happen outside the bounds of a `QuadTreeLayout`.
PanAndZoom.cs	Camera behavior for zooming and panning.
Parallax.cs	Creates a parallax by moving an object slower/faster than the camera it is seen through.
Periscope.cs	The application controller.
Prettyprinter.cs	Prints the current object hierarchy to the console.
PrettyprinterEditor.cs	A custom inspector for the Unity Editor, capturing prettyprinter output.
QTreeCellDropHandler.cs	Allows cards to be dropped into trees.
QuadTree.cs	A custom quadtree data structure.
QuadTreeLayout.cs	Layouts `RectTransform` objects in a quadtree subdivision pattern based on their relative position.
ScaleToMatchSize.cs	Scales a `RectTransform` object to match the size of another.
Session.cs	A serializable representation of the application state.
SessionFilePresenter.cs	Visual display of the current session file.
SidebarController.cs	Controls the sidebar panel.
StackCellDropHandler.cs	Allows cards to be dropped into stacks.
StackController.cs	Controls the behavior of a stack.
StackFlow.cs	Allows stacks to be browsed like a deck of cards.
StackHotkeys.cs	Allows a stack to respond to certain hotkeys.
StackPresenter.cs	Manages the visual display of a stack.
StackShuffle.cs	Allows a stack to be shuffled.
StringExtensions.cs	Various utility methods for strings.
SystemInfoPresenter.cs	Displays statistics about the current application state (memory usage, frame-rate, cpu load).
TextureScale.cs	Bicubic texture scaling.
Tools.cs	Manages the tool mode (edit, select, draw).

C Explorative Research Interview Responses

First pass compression of interview responses obtained during explorative research in chapter 3.5. The items below do not represent actual expressions used by participants but instead the general messages that were expressed. The items are in chronological order. All entries include references to the identity of and time codes in the audio recording they were taken from. The items have not been translated.

P1 02:00 Methoden zur Innovation bedienen kreativen und nicht-kreativen Anteil im Prozess

P1 03:30 Konstellation der beteiligten Akteure, andere befähigen

P1 05:00 Ideation findet bedarfsorientiert, kollaborativ statt, zielt auf Umsetzung; Themen Experten vs. Ideation Experten haben jew. unterschiedliche Einflüsse

P1 05:00 Begriff der "Idee"

P1 05:30 Aus sparks ableiten wie Technologie funktioniert und was sie ist (Teilnehmer ohne Vorkenntnisse)

P1 06:00 Beobachtung der Gruppe -> Ableitung einer Einschätzung bzgl. Qualität/Sinnhaftigkeit/Wirksamkeit/Pertinenz der Sparks

P1 07:00 Erwartung an Sparks: Heterogenität, tiefere Einblicke, Information, Orientierung

P1 07:30 *Wunsch: Bei Erstkontakt mit Sparks spontane Sortierung in vordefinierte/allgemeine Kategorien wie Unsinn/Interessant/Langweilig die erste Eindrücke repräsentieren.*

P1 08:30 *Manche Sparks nervig weil offensichtlich.*

P1 09:30 Motiv der Spark-Autoren: Clickworker, Horizont erweitern/Nachdenken ist anstrengend, folglich Qualität fragwürdig.

P1 10:30 Schneller Takt des Ideation-Prozess positiv, fördert fortschritt, Fokus, Produktivität

P1 11:00 Sparks rolle initial: Ein Bild der Technologie machen (Teilnehmer kennt Technologie nicht)

P1 11:00 *Wunsch: Mehr Interaktion zw. Teilnehmern, d.h. Ping-Pong*

P1 12:00 *Wunsch: Kritische Nachfrage, Dialog*

P1 13:30 Divide & Conquer der Sparks: Schaffung von Bereichsexperten bzgl. Verteilung von Spark-Untermengen auf Teilnehmer

P1 14:00 Erschöpfende Diskussion, erschöpfende Extraktion von Ideen (Wie viel ist genug/ausreichend?)

P1 14:00 Begriff "Fruchtbarkeit"

P1 14:00 *Verlust von Ideen/Einsichten/Perspektiven über Zeit wenn kein Ausdruck/Wiederholung/Diskussion*

P1 15:30 *Prüfen von Ideen durch Expertenlinsen (Andere Teilnehmer): Gegenseitig, Dialog, Zeitversetzt*

P1 16:00 Diskussion (Konvergent) kann leicht unproduktiv werden, braucht Struktur. In Divergenter Phase verbieten(?)

P1 16:30 *Mehrere Cluster-Ebenen, d.h. vorgegebene und emergente Cluster, z.B. Quatsch/Off-Topic, Eigene und Cluster von Anderen. Bietet Reibungsfläche.*

P1 17:00 Zeitmanagement

P1 17:00 Ausreißer-Sparks können bei erstem durchlauf und Segmentierung der Sparks auf Teilnehmer nicht identifiziert werden (Welchen Einfluss hat das?).

P1 18:30 Clustering-Runden, ggf. ineffektiv wenn falsche Strategie/Cluster.

P1 19:00 Clustern widerspricht "Offenheit", d.h. Kontraktion/Konvergenz nicht zu früh einleiten.

P1 19:00 *Cluster Strategien für "Wonach schaue ich?". Kann jew. ins leere führen. Deshalb mehrere Runden vorsehen mit Rollback/Snapshot/Wiederverwendung/-Zeitmanagement*

P1 19:30 *"Erst mal alles sehen" (Sparks) und beobachten was "triggert". Daraus Strategie ableiten (emergent), d.h. Trigger-Ebene einführen.*

P1 21:00 *Wunsch: Markieren, hervorheben und plakativ machen um Gleichförmigkeit der Vorlage/Sparks aufzuheben, als Bild zusammenbauen.*

P1 23:00 *gegenseitiges Idee-Vorstellen: konkrete Bilder entstehen*

P1 21:30 *implizierter filter vs. expliziter filter*

P1 24:00 jeder hat sein Steckenpferd (Framing)

P1 27:30 *Digitales Hilfsmittel wird Synonym mit Suchmaschine gesehen.*

P1 25:00 *Outcome hängt von Framing ab, insbesondere bei inkrementellem Vorgehen. Ggf. bewusst iterativ Framing ändern.*

P1 27:00 Mehr Gelegenheit für Kontext + Research bevor Idee entseht

P1 27:30 Suchmaschine nur interessant mit viel mehr Zeit für research zwischen Iterationen/Phasen

P1 30:30 Methodensicht: in divergenter phase sollen konvergente impulse unterdrückt werden. Und umgekehrt. Aber, inwieweit hilft Bruch dieser Regel beim jew. Phasen-Ziel?

P1 31:00 Methodische Anleitung gut, soll tatsächlich aber Flexibilität erlauben: professionelle Umgebung wichtig damit alle Teilnehmer sich des Maßes und der Qualität einer Störung bewusst sind und nicht das Ziel aus den Augen verlieren

P2 03:30 Innovationsprozess: diverse Hintergründe der Teilnehmer; gemeinsam Antwort auf eine Frage entwickeln

P2 03:30 Zwei unterschiedliche Ausprägungen von Innovation: Nutzergetrieben (von aussen) und aus intimem Verständnis einer Sache heraus (von innen).

P2 04:30 Von innen ist eher disruptiv, "Hat vorher keiner gesehen", "gab vorher keinen bedarf", "Beispiel iPod".

P2 05:00 Dichotomie muss nicht unbedingt so sein. Disruption kann aus tech., ökon. und soz. Verständnis einiger Experten abgeleitet werden. Typisch für "von innen".

P2 07:00 Einflüsse und Rolle der Sparks: Begriffe, Handlungskategorien, Material, Eigenschaften, Umgebungen aufzeigen, d.h. Metaebene konstruieren auf der dann eigene Ideen entstehen.

P2 07:30 Sparks = Anlass anzufangen: Reibungsfläche, Futter, Einzelne Sparks an sich nicht wirklich relevant.

P2 08:00 Diskussion über Metaebene ist effektivste Komponente der Auseinandersetzung.

P2 08:30 Inhalte der Sparks vielleicht relevant für Outcome. Hypothese: Systematische Heterogenität effektiv? Schwierig?

P2 10:30 Zeitdruck durch Beobachtungssituation, Synthetische Situation, Stress, Will zum Ziel kommen, Verkürzt, Qualität evtl. besser mit mehr Zeit.

P2 11:00 Team kennt sich, das beeinflusst Art und Effektivität der Zusammenarbeit.

P2 11:30 Neugier auf Input von Anderen; Jeder hat andere Qualitäten/Sichtweisen/Linsen/Perspektiven

P2 12:30 *Rollen anderer Teilnehmer: Ausbrechen aus dem Prozess/Metaebene öffnen, Offenheit/Inspiration, Durchdacht/Feinheiten, Nutzerperspektive*

P2 17:00 *Physische Situation beengt, "im Weg stehen". Impuls: "An der Wand machen mit mehr Platz".*

P2 18:00 *Aha!-Momente wenn gute Cluster von Anderen auftauchen. Hilfe für Ordnung im eigenen Kopf.*

P2 19:30 *Ideen veränderbar machen: streichen, hervorheben, beschriften*

P2 23:00 *Eigene Idee niederschreiben/Worte finden*

P2 23:30 *Für sich arbeiten (um input zu verarbeiten)*

P2 24:30 Beeinflussung durch externe Faktoren: Wiederholung, Vorwissen, ...

P2 26:30 *Verfügbarkeit von Materialien, Anregungen, Inputs*

P2 27:30 Gedanken suchen stets Verbindungen in der Pluralität der Einflüsse. D.h. alles verbindet sich irgendwie, maximale Vielfalt ist wünschenswert.

P2 28:00 In Bewegung Ideen finden, Begegnungen mit Dingen ausserhalb des normalen Alltags (Kontrast herstellen, dadurch neue Assoziation anregen)

P2 28:30 Ausblenden wenn etwas zu viel wird.

P2 29:30 *Schreiben, Hinlegen, Zeichnen, Entscheiden, Codieren sind essentielle Denkprinzipien.*

P2 29:30 *Automatik die den Prozess dokumentiert/Aufzeichnet, ggf. Elemente automatisch manifestiert ist nicht effektiv, sogar verwirrend. Manuelle Codierung ist wichtig für Reflexion und und Modellbildung.*

P2 30:15 Situation der Ideenfindung; Nutzung des Ortes;*Ordnung im Raum, Handlung/Geste, Entscheidung, Abstrahieren, übersetzen*

P2 31:30 Anregungen durch Materialien

P2 32:30 "Beispiele die ich immer zur Hand habe", Prinzipienkisten, Material, Farben, Bilder, ...

P2 34:00 Anregungen jederzeit greifbar. Viel platz schaffen. Beschränkungen entfernen. "Direkt auf den Tisch schreiben". (Gelegenheit radikal zu sein).

P2 36:00 *Blockaden; Zeitbegrenzung, Anregungsbegrenzung, Zu klein, kein Zeichnen möglich, Idee passt nicht in formale Vorgaben und findet dann keinen Ausdruck.*

P2 37:30 *Introspektion; Ausformulieren, Worte wählen, Zeit allein*

P2 39:00 Beharren auf eigenen Einsichten vs. Offenheit für andere Perspektiven; Gruppendynamik aus individuellem Vor-Clustering

P2 41:00 *Wunsch: Clustern mit Strategie vs. ganz offen; Allg. gültige bzw. klassische Achsen/Kategorien erlauben es weiße Felder zu erkennen, vgl. Landkarte; Häufungspunkte machen Topologie sichtbar*

P2 43:30 *"Kosmos von Dingen die ich abfrage"; Repräsentanten die sichtbar machen und erinnern (Kategorien)*

P2 45:00 *Fragen die immer wieder vorkommen; Grundprinzipien der Welt ("Magnetismus"), Linsen/Mechanismen/Wirkweisen/...*

P2 46:30 Studi-Meinung/Angst: Methoden grenzen ein;

P2 47:30 *Vorgedachte Kategorien und Strategien verhindern Ideen. Erfahrungswert. Ausprägung unterschiedlich je nach art der Innovation (innen/aussen/Thema/Kontext).*

P2 49:30 Verhindert innovation "von innen". Hemmt unerwartete Assoziationen, verdeckt weiße Felder die nur bei ungewöhnlicher Betrachtung/Perspektive sichtbar sind.

P2 50:30 Angeleitete Innovation "von aussen" zielt mehr auf Verbesserung. "von innen" und frei mehr auf Synthese von Experten und disruption.

P3 02:30 Begriffe: Projekt planen, gemeinsame Ziele, Zeitplanung, deskriptiv, chance, tatsächliche Innovation, Ziel, innovation = Nutzbarmachung einer Erfindung.

P3 04:30 Begriffe: Rückblick, Struktur, Messbarkeit, Weitergeben, Aufarbeiten, Idealtypisch, Prozessplanung

P3 05:30 Gruppendynamik, Innovationsfreundlicher Kontext

P3 06:30 Framing-Iteration, Klima-Iteration

P3 12:30 Begriffe: Konstrukte, Cluster, Begriffe, Aussortieren, Seltsame Idee zu etwas nützlichem weiterentwickeln, Diskussion von Meta-Ebenen, etwas neues entsteht, Auswahl und Genese, konvergent und divergent.

P3 13:30 Sparks beeinflussen; Framing, ggf. fixation durch Wiederholung; pos/neg Qualität der Einflüsse abwägen; möglicherweise andere Ideen mit anderen Sparks; Alternative Sichtweisen aus den Sparks ableiten, was zu neuen Ideen führen kann; Neue Kontexte aufspannen;

P3 14:00 *Sparks werden zu Opportunity Areas zusammengeführt*

P3 16:00 Idee ist ohne Kontext/Anwendung wertlos (Zusammenführung von Idee/Spark und Kontext/Anwendung)

P3 17:00 Gefahren: Kleinteilig, Zu speziell, Zu Nische, zu geringer Wert, zu hohe Kosten, zu kleines Volumen, fehlende Complements, fehlende Vernetzung, fehlende Infrastruktur, fehlende Bereitschaft/Akzeptanz

P3 17:30 *Momente der freien Assoziation (Spontane Verfügbarkeit einer Reihe von Reibungsflächen um Brücken zu bauen bzw. Assoziation anzuregen)*

P3 18:00 *Komisch/Absurd/Schräg/Irritierende Sparks sind auffällig und merkbar; werden wiederholt betrachtet.*

P3 19:30 *Begriff: Effektivität*

P3 20:30 *Auch abstruse sparks verknüpfen sich (Hirn versucht Dissonanz irgendwie aufzulösen). Sind sogar besonders wirkungsvoll gegenüber gewöhnlicheren.*

P3 21:00 Aufgabe/Kontext/Ziel; Auf Abwege achten. Insbesondere: Fixation durch Sparks die sich wiederholen.

P3 23:00 Begriff: "Dedektieren"

P3 23:00 *Effektiv; Kontexte wechseln*

P3 25:30 Gruppensituation; kann hemmen, persönliche Situationen können irritieren, Teilnehmer können dadurch verloren werden. Abgrenzung Produktiv/Konstruktiv.

P3 27:30 *Gruppensituation; Ping-Pong*

P3 28:00 *Wunsch: lieber weniger (als 5) Teilnehmer*

P3 29:00 *Konkretes Cluster-Ergebnis ist nicht wichtig, Dialog/Diskussion auf dem Weg ist der eigentliche Wert.*

P3 29:00 Mehr Zeit, je nach Zielstellung

P3 30:00 Ideation Idealtyp: Eher iterativ in kurzen Etappen, 45 min.

P3 32:30 Allgemeine Kategorien nicht hilfreich, haben keine Bedeutung ("für mich").

P3 36:00 Situation: Tisch Setup gut.

P3 37:00 *Notizen hierarchisch auf Cluster-/Spark-/Meta-Ebene möglich machen. Überlappende cluster ermöglichen.*

P3 37:30 Situation: Zwischenstand dokumentieren (nützlich wofür?)

P3 40:30 Haufen vs. ausbreiten; Mehr Platz nötig, sonst: unstrukturiert, Verlust von Zwischenständen und Übersicht.

P3 41:30 *Wunsch: Pausen; Gruppendynamik zurücksetzen, Ideen reifen lassen, durchatmen*

P3 42:00 Wunsch: Tech-Experte

P3 42:30 Timing von Einflüssen und Möglichkeiten (Hilfsmittel) wirkungs-/Zielgerichtet steuern/planen

P3 45:00 *Assoziations-/Ideenketten; zufällige impulse von anderen, serendipity*

P3 47:30 *Titel/Formulierung einer Idee klärt und leitet weiter*

P3 48:30 Transformation einer Idee in diverse Kodierungen, diese dann wieder kontextfrei als Spark für die ff. assoziative Suche benutzen; Form syn., Wort syn., Konzept syn.; inspirierende Titel.

P3 49:30 "Telefon-Joker" (Technologie Experte), jedoch limitiert in der Verfügbarkeit (Reibungsfläche)

P3 50:00 Timer für Zeitmanagement um Überblick zu behalten.

P3 51:30 Gefahr sich in Diskussionen zu verfangen

P3 53:00 Auch komische un 1-Wort Sparks sind wertvoll.

P3 55:30 Begriffe: Legitimation, Anwendungsgrund

P3 60:00 *Ideenfindung; Kommunikation mit selbst, Skizze oder vorgestelltem Framing; Mental walkthrough; Idee jemand anderem erklären, damit eigene Missverständnisse klären.*

P3 61:00 In Realität überführen, dadurch hinterfragen (Idee wird erst in konkreter Anwendung/Realität gut/schlecht);

P3 62:00 "jeder Trigger ist erstmal OK".

P4 03:30 Methoden und Menschen zusammenbringen.

P4 04:30 Innovation, Besser machen, Neuheitsgrad, Verorten, Wissen, Manifestieren, Problem

P4 06:30 Disziplinen/Felder zusammenbringen.

P4 07:00 Begriffe: Marktfähig, realistisch; Immer mir Wirklichkeit in Einklang bringen.

P4 09:00 *Motiv wichtig für Reflexion*

P4 11:30 Clustering elementar für Sense-Making; Teil jedes Denkprozesses (hier syn. mit Input-Filter); Jeder benutzt das; Sortieren, Ordnen, Kategorien bilden; Notwendige Komplexitätsreduktion

P4 12:00 Begriff: Kontext

P4 12:30 Explizites Clustering dient der Kommunikation, kann aber nicht 1:1 das abbilden was der Autor im Kopf hat. (Potential für Nachfragen, Diskussion, Reflexion)

P4 13:30 Codierung einer Idee ist schwierige Aufgabe; Macht Fehler und Qualität sichtbar

P4 14:00 Mediums-Wechsel verändert die Information

P4 14:00 Begriff: Schwarm-Intelligenz

P4 15:30 Fehlt: Visuelle Komponente, visuelle Trigger, alles ist Text, wir sind Designer. (Recht eingeschränkte Sichtweise was ein "Designer" ist)

P4 16:00 Disziplinen haben unterschiedliche trigger Mechanismen

P4 17:00 Visuelle Komponente; Repräsentanten (icons/Titel/Token) müssen von der Gruppe (schnell) gelernt werden; Teilnehmer können ggf. nicht Zeiteffektiv Repräsentanten anfertigen, der manuelle Prozess ist jedoch elementar und kann nicht automatisiert werden ohne Wert zu verlieren.

P4 17:00 Begriff: Visuell

P4 17:30 Rolle im Team; Übersetzer/Formulierer

P4 19:00 Übersetzung immer hilfreich als Reflexionsfläche; Gedächtnis stützen (Erinnerung), Netzwerk stricken, Umschärfe der Semantik nutzen.

P4 20:30 Symbol/Sprache Kombination um Brücke zwischen verschiedenen Denktypen zu schlagen

P4 23:30 Bildungshintergrund bestimmt teilweise Fähigkeit zum Wirksamen Einsatz und Bearbeitung von bestimmten Aspekten/Themen/Einflussfaktoren (vgl. Ingenieur vs. Kulturwissenschaftler)

P4 26:00 Aufbereitungsproblem; Sparks können nicht quer gelesen bzw. überflogen werden.

P4 27:30 *Experten fällt es schwer Quatsch "auf den Tisch zu legen" der aber signifikant als Inspiration effektiv ist; Wortlaut der Sparks wird schnell zurückgelassen/abstrahiert, dient also nur als Ausgangspunkt. Sparks sind Selektion von Triggern die relevant für das gegebene Thema sind.*

P4 29:30 Clustering "heute" = Download von Info mit spontanem Vor-Clustering/Vorsortierung (Quatsch, Interessant, ...)

P4 30:30 *Sparks schwer aufzunehmen, Viele Details ohne Nutzen (Was ist nützlich?)*

P4 33:00 Clustering führt zu Verständnis und Überblick über das Thema; Sense-Making, learning

P4 32:00 keine bewussten aha Momente

P4 36:00 Methode/Material bestimmt outcome (?)

P4 37:00 Clustering "gerne" ein ganzer Tag, 1-2 Tage Ideation, nicht nur 45 min, Zeitdruck wird neg. empfunden.

P4 37:30 Ping-Pong zw. Gruppen- und Einzelarbeit; Mit mehr Zeit auch Einbindung externer Technologie Experten.

P4 40:00 Timing hat Bedeutung/Relevanz

P4 40:30 Zeitdruck lässt keinen Raum Blockaden aufzulösen

P4 41:30 Methodische Abzweigungen; "Lass uns zu X nochmal clustern".

P4 42:00 Farbige Markierungen wurden nicht genutzt (Autor Tracking), ggf. im Rückblick interessant, jedoch als Anregung Bedeutungslos; Ggf. nützlich für Rekonstruktion.

P4 49:30 Bsp. Anregungen als Hilfsmittel; Den thematischen Gegenstand betrachtbar, erfahrbar, manipulierbar machen; Bezug herstellen; Eigenschaften erfahren; Anschauungs-Kit;

P4 55:30 Konzeptanalyse; Tools hilfreich; alternative Suchstrategie als die eigene; Tool ist Ersatz für Team-Mitglied und dessen andere Perspektive; Frage der Legitimität? Ggf. bewusst konträr.

P4 57:00 Smartboard-Desaster. Tech. kann die Komplexität des Prozesses nicht abbilden bzw. ihre gerecht werden. Praktisch nicht nutzbar. Flexibilität von Post-It apps nicht ausreichend.

P4 57:30 Umfunktionieren/Hacking von Tools durch Menschen; "Digitale Symbole zu rigide"; Embodyment (Kommunikation durch Geste) fehlt. 3D Raum fehlt; damit Betonung des Ortes über Geste schwierig.

P4 58:30 "Eingeschränktes Vermögen diese Daten im Speicher zu halten"; Überblick, Working Memory, Wiederholung

P4 59:30 Gleichzeitigkeit; aktiver Speicher besser nutzen; Detail Reduktion

P4 62:00 Manifestation durch Aufschreiben und dadurch "Wichtigmachung" bzw. Hervorhebung/Markierung; Titel

P4 62:30 Regelwerk für Modellierung von Begriffen die manifest werden; "Öffnend"

P4 64:30 Analyse/Ziel Klären; Rahmen aufstellen; Freiräume abstecken; Bauchgefühl; Leitplanken (Zeit, Budget, Thematik, Regeln, Ethik)

P4 65:00 Methode hat Freiraum; Bewusst aufbrechen

P4 65:30 Methoden-Formulierung ist Notwendig für Kommunikation. Nicht jedes Wort ist hergeleitet und gleich bedeutsam. Methoden kombinieren.

P4 66:00 Interdisziplinäre Gruppenarbeit

P4 66:30 Idee hat was mit Bewegung zu tun. Ideal nicht im abgeschotteten Raum sondern in der Welt. "Prototyping am Ort des Problems nachdenken."

P4 71:00 Ideation-Tools sind schwer standardisierbar, Aufgaben/Probleme sehr individuell jedes mal, Kein Idee-Baukasten vorhanden -> kreativ

P4 72:30 Störung/Intervention mit Prototyp und Reaktion beobachten; Anregung, Reflexion

P5 02:30 Begriffe: Ideenraum, Möglichkeitsraum, Abstecken, Gedachte Barrieren, Materialien, Rahmen, Zwang auszubrechen, kategorisierbar, zulassen, Puzzle.

P5 03:30 Bewusst gesetzte Schranken machen Ideen kategorisierbar relativ zu den Schranken, erzeugt ein Verlangen auszubrechen, Vertrauen in Auftreten unvorhergesehener Kombinationen,

P5 04:30 Erkannte Probleme und Aspekte separat und isoliert durchdenken; bewusste limitation.

P5 06:00 Clustering dient dazu Dinge festzuhalten, z.B. in einem flüchtigen Gespräch, mittels Zuweisung von Titel, Kategorie, Zeichnung, Repräsentant, Bild, "Lesezeichen".

P5 08:00 Persönliche, invariante Sammlung kategorisierter Bilder als Anregungs-Geber (Sparks)

P5 08:30 Sparks beeinflussen die Perspektive

P5 09:00 "Was sind das für leute?" (bezogen auf Spark-Autoren); Beruf, Lebenssituation, Ort, usw.

P5 10:00 Sparks sind nur Anlass, "Rest passiert im Gespräch"; Sparks schnell verlassen; Sparks aussortieren

P5 11:00 Sparks markieren, verändern, hervorheben

P5 12:00 gute sparks sind präzise und konkret, geben eine Startpunkt um abzuwandern

P5 13:00 Seltsame Sparks helfen eigene Muster aufzubrechen

P5 15:00 Cluster in erster Runde unsicher; im Austausch mit Gruppe entsteht besserer Überblick und damit bessere Einordnung der eigenen Cluster; dadurch mehr Sicherheit.

P5 19:30 Karten, Stifte, Tisch; "mehr braucht man eigentlich nicht".

P5 19:30 Hilfreich: Ehrfurcht vor Sparks (sorgfältig vorproduzierten Anregungs-Gebern) verlieren.

P5 19:30 Farbenblindheit

P5 21:30 Ordnungsmuster oder -regel gibt Anlass zur Reflexion

P5 22:00 Gutes Gefühl mit Sicherheit "Trash" auszusortieren.

P5 22:30 Zerlegung von Sparks auf interessante Teile

P5 23:00 Wunsch der temporären Isolation (anders als manche Anderen)

P5 24:00 "Wer sind die Autoren?"

P5 25:00 Idee braucht Anwendungsfall (sonst wertlos)

P5 35:00 Clustering und Ideation sind separate Phasen; Ping-Pong zw. Einzelarbeit und Gruppenarbeit.

P5 37:00 Zuordnung von Sparks zu Clustern oft nicht eindeutig. Besser nicht zu viele Sparks. In erster Runde der Sortierung gleich Favoriten markieren.

P5 43:00 Begriff: Gestaltungs-Ingenieur

P5 43:30 "Gibt Hebel in der Gestaltung die man wie ein technisches Instrument nutzen kann"

P5 45:00 Beiträge in Gruppenphase "lesbar" für andere machen. Klare Bilder, starke Wörter, allgemeine Verständlichkeit.

D Prototype Evaluation Interview Responses

First pass compression of interview responses obtained during the prototype evaluation in section 4.6. The items below do not represent actual expressions used by participants but instead the general messages that were expressed. The items have no particular order. The items have not been translated.

XR man kann ohne strategie zum ziel kommen.

XR texte brauchen viel zeit zum lesen.

XR blaues markierungs-overlay macht bild schwer zu erkennen (umständlich)

XR wirkt sehr strukturiert

XR material stapel haben die ideenfindung positiv beeinflusst.

XR man kann nicht gut anbauen, insofern gibt die software etwas vor

XR wurde als sehr produktiv empfunden

XR viel cooles material

XR besser als nur ein ordner mit bildern und generisches tool zum arrangieren

XR outcome basierte vollständig auf den karten.

XR durch die ästhetik fühlt man sich als kreativer verstanden und ernst genommen

XR die aufgabe war so gestellt dass sie gut mit der software bearbeitbar ist

XR texte wurden ignoriert weil das lesen zu lange dauert, kurze texte wurden trotzdem genutzt.

XR mit inhalten (text/ bild) konnte produktiv umgegangen werden.

XR vermisst, ctrl-z um vergessenen vorigen zustand wiederzufinden...

XR ctrl-z als explorations-mechanismus (korrektur von fehlern wurde im zusammenhand mit ctrl-z nicht erwähnt)

XR wirkt "template-mäßig", wie ein aufruf: man soll das jetzt befüllen.

XR tool ideal um erstmal gedanken zu sammeln, ohne vorher überlegungen zu struktur anzustellen. inutitiv. sich treiben lassen.

XR konnte ein eigenes muster intuitiv entwickeln

XR frage ob wohl eine eigene aufgabe gut lösbar ist mit dem tool

XR glaubwürdigkeit gesteigert duch gute gestaltung.

XR bin schnell zu etwas gekommen (wegen stapel)

XR analoge methoden sind sehr unmittelbar

XR eigene methode: durchstöbern von material um zu erkennen was man selbst gut findet (browsing/ exploration)

XR man kommt auf neue wege auf die man mit papier und stift nicht gekommen wäre

XR tool hat gefolfen die inhalte nutzbar zu machen.

XR 4-teilung sehr dominant

XR material: bilder sind in kurzer zeit zugänglicher als texte

XR ästhetik erstmal nicht mit kreativ-tool assoziiert

XR selbstkontrolle (um eigenschaften phys. obj. mit software zu simulieren) negativ bewertet.

XR unmittelbares gefühl der haptik und körperlichkeit (software setzt das nicht um)

XR das erstellen physikalischer objekt hat "kosten" über die man sich gedanken machen muss, software hat das nicht, wirkt dadurch befreiend (was allerdings als negativ empfunden wurde)

XR digitale tools verleiten zum "alles speichern" und horten.

XR würde software für eigene arbeit/ projekte gerne verwenden

XR physikalität zwingt zu finalen entscheidungen.

XR software gibt methode vor.

XR 4 teilung führt zu entscheidungszwang, wurde positiv bewertet.

XR scheint sich eine signifikante trennung von notizen (nebenbei) und der arbeit im tool (produktivität)

XR auch ein präsentations-tool, vordergründig jedoch ein persönliches tool.

NR hat spaß gemacht, aufgabe war spaßig

NR eigene aufgabe muss evtl. mit klarem verwendungsmuster des tools angegangen werden.

NR shortcuts ähnlich cad software, creative cloud und actionbar in computer spielen

NR material scheint passend für die aufgabe vorbereitet

NR software hat nicht die selbe autoriät wie phys. objekte da man weiß dass sie nicht den selben beschränkungen unterworfen sind. (nicht legitime leistungs-verweigerung)

NR ein stift setzt intention direkt um.

NR beschreibt drag & drop als eine geplante handlung die ausgeführt wird.

NR wert der physikalität von objekten: hilft beim gefühl etwas geschafft zu haben, sichtbar, greifbar, physisch da, unmittlebar.

NR methode/muster des STACKTREE wurde positiv/ freundlich bewertet

NR viele überlegungen wie mit der 4-teilung umgegangen wird. widerstand sofort erkennbar. wird allerdings nicht als anstrengend empfunden

NR gehin aufteilen in 4. diese dann wieder teilen für details.

NR wünscht sich eine funktion zum zeichnen (nicht bestandteil der eval)

NR software ideal für schnelles intuitives "brainstorming".

NR ästhetisches interface.

NR lüfter wurde störend empfunden

NR leere session ggf. nicht produktiv

NR direkter nutzen der software über andere methoden nicht direkt identifizierbar (kein vergleich)

NR bezüge zw. karten nicht explizit

NR bildmaterial hat ersten eindruck (software für notizen) aufgebrochen.

NR vorgefertigtes material ist wichtig und nützlich

NR kein stock material: ist schön

NR zeichnen evtl. mit ipad stift gut vorstellbar

NR vier felder um "mental spaces" zu machen

NR "man kommt schnell rein"

NR software für textuelle ideation nicht interessant genug, für visuelle projekte sehr hilfreich.

NR nutzung als text oder collage tool, bzw. mischung aus text und assoziativen bildern

NR blau gibt kein gefühl von "kreativität" (kein vergleich)

NR vorschlag: zeichnen als separates "notepad".

NR für präsentation andere form als während dem privaten denkprozess

NR vorschlag: eigene bilder (geht schon)

NR mit der zeit dann doch nicht komplex

NR kein impuls alternativ zu einem stift zu greifen

NR hat geholfen einen einstieg zu finden

NR content anregend

NR abspalten von "clustern" die "fertig" sind um "aufzuräumen"

NR betrachtet baum als separates funktionales modul

NR eigene ordnung finden

NR mag muster » erster eindruck erzeugt legitimität

NR zeichen-funktion kritisch in einem längeren, professionellen prozess

NR vermisst funktion zum zeichnen

NR für präsentation der ergebnisse würde die struktur umgeordnet um eine didaktische struktur / narrative abzubilden

NR erster eindruck: hier kann man notizen reinschreiben

NR erstmal komplex

NR intuitive auswahl der bilder

EN stacks ggf. sinnvoll bei längerer nutzung

EN hatte anfänglich die idee dass drag & drop in den stack die karte dupliziert (war kein problem)

EN stacks verstanden, bedarf nicht erkannt (im 15 min task)

EN interpretiert hintergrund-muster als hinweis dass dies ein kreativitäts-tool ist

EN stack anzahl beliebig (evtl. nicht vorgeben, evtl. nutzer selbst anlegen lassen)

EN mag formalisierte design-prozesse nicht

EN hintergrund-muster: "du musst mir nicht sagen dass ich hier was kreatives mache", "herablassend"

EN erster eindruck hintergrund-muster überflüssig

EN sympathisch dass die software mich machen lässt was ich will. "no fuss"

EN tool bietet eine gut abgestimmte palette von möglichkeiten (sich auszudrücken)

EN würde das tool tatsächlich nutzen.

EN überrascht dass die nutzung des tools neue ideen erzeugt hat (zusätzlich zu schon existierenden überlegungen zum task kontext)

EN verwendung der stack-karten ist freiwillig (nicht erzwungen durch eine methode), dadurch willkommen.

EN tool ist für kreative profis, schaffende im weitesten sinne, auch forscher

EN hat mich positiv beeinflusst

EN später: viele kontrollen sind kein thema mehr

EN erster visuelle eindruck (komplexität) hat sich nicht negativ bestätigt

EN hat mich nicht behindert

EN erster eindruck: schön aufgeteilt

EN später: nur wenige kontrollen benutzt

EN erster eindruck: viel auf einmal

EN erster eindruck file-optionen (nicht test relevant) auffällig

EN tool hat resultat beeinflusst und prozess durchaus gut unterstützt.

EN hauptsächlich benutzt: zoom/pan, drag & drop editieren, duplizieren » kontrollflut weg

EN qualität und auswahl des stapel- materials hat viel einfluss.

EN nicht primär ein tool für präsentation, kann aber eines sein, und würde sinn machen

EN slogans im stack würden in einem normalen umfeld eher abweisend wahrgenommen.

EN keine alberne methode die nichts bringt

EN sofware wird teil einer schaffenskette

EN vorschlag: form und farbe von karten ändern

EN vorschlag: vorgefertigte karten können verändert werden.

EN tool ist relativ restriktiv aber innerhalb der beschränkung sehr flexibel

EN auffällig beim ersten eindruck: kontrollinfo in der sidebr

EN favorisiert produktions-tools die zugang zu gestalterischen fähigkeiten ermöglichen

EN mag kein design thinking

EN prinzipielle ablehnung von design methoden.

AH wünscht sich materialien die das "state of the art moment" einfangen

AH vorschlag drag & drop

AH gefällt: mischung aus limitierung (durch struktur) und vielfalt

AH double diamond

AH bewertet 4-teilung als universell anwendbar

AH vorschlag: export der sich an papierformate anspasst.

AH wünscht sich mechanismus um beziehungen explizit zu machen um metastrukturen einzufangen. zb farben

AH wünscht sich eine export funktion.

AH bewertet nachvollziehbarkeit von zustandsänderungen während paar-interaktion als gut.

AH inital ungewohnte nutzung.

AH beszeichnet sich als nicht ad-hoc kreativ.

AH wünscht sich export von teilen einer session

AH sieht allgemein schwierigkeiten bei aufgaben die sich mit schwer darstellbaren konzepten beschäftigen (z.b. werte)

AH konnte duch das vorhandene material spontan kreativ sein

AH bewertet die materialien als erstaunlich wirksam um unerwartete ideen zu entdecken

AH sieht sehr textbasierte sessions als schwieriger als bildbasierte

AH fühlt sich von der testsituation wenig beeinflusst

AH bemerkt keine explizit negativen aspekte

AH bewertet struktur wiederholt sehr positiv

AH hat durch das zeitlimit versucht relativ schnell zu arbeiten

AH entdeckt tendenziell für alle bemerkten schwierigkeiten die aus der geometrie entstehen eine mögliche lösung.

AH spürt drang leerräume zu nutzen bzw. zu vervollständigen.

AH wünscht sich materialen zu briefing hintergrund.

AH kann sich teamnutzung vorstellen

AH vorschlag: material stapel mit denen sich die diversität der ideenfindung steuern lässt.

E Evaluation Protocol

Periscope Evaluation Protocol

Participant ID	Software Version	Date

1

Introduction

Today you will participate in the demonstration and evaluation of a prototype software called **Periscope**.

Please understand that we are **evaluating the software** and not you. Any and all confusion, misunderstanding and mistakes are the fault of the software and the design of this evaluation process. It is our goal to uncover as many issues with the software and the evaluation process as possible.

The evaluation will be conducted by a moderator/observer you can **ask questions** at any time. You are encouraged to **talk aloud** while interacting with the software, commenting on what you aim to do and which issues you encounter.

Your interactions with the software and commentary will be **recorded**. In particular the screen, keystrokes, mouse-movement, environment audio and your visible reactions.

Beginning each step of the evaluation you will receive written **instructions**. These instructions will also be read out to you. All instructions are contained in this manual. **Please do not read ahead. You will be asked to turn pages at appropriate moments.**

The evaluation will take approximately **45 minutes**:

Minute **0-5**	Software **demonstration**	(5 min)
Minute **5-10**	Casual **familiarisation**	(5 min)
Minute **10-25**	Solving a creativity **task** with the software	(15 min)
Minute **25-30**	Usability **questionnaire**	(5 min)
Minute **30-45**	Semi-structured **interview**	(15 min)

Checklist

Yes	No	
☐	☐	I understand the instructions given above.
☐	☐	I am able to comfortably observe the software demonstration about to begin.
☐	☐	I have signed the consent-form.

Before we continue, do you have any questions?

2

Pre-Assessment (Answer immediately)

How comfortable do you feel performing an ideation task with the software? Low ☐☐☐☐☐☐☐☐☐ High

What is Periscope?

Observe: Periscope is a software that helps you think in an explorative way. To do this, Periscope provides an environment that facilitates innovation: The discovery of configurations of elements that open new doors into adjacent possibilities.

Deliberate effort towards innovation aims to discover these doors and to walk through them. This works best when the search for them happens at the edge of chaos, where chance connections between elements may be formed, that persist just long enough to be recognised as useful building blocks.

In our case, deliberate innovation of manmade things this recognition of useful building blocks is often given by a local network of people, informally sharing and discussing their work with each other, challenging assumptions, offering interpretations and most of all, inspiration: passing around ideas as building blocks to one another.

Periscope aims to provide an environment that offers a primordial soup of elements, chance connections and the opportunity to discover useful building blocks towards a stated purpose.

Before we continue, do you have any questions?

User Experience Consent Form

Participant ID	**PARTICIPANT PRIVACY: THIS PAGE MUST BE KEPT SEPARATE FROM ALL OTHER MATERIALS ASSOCIATED WITH THIS ID. THIS SEPARATION MUST BE PROTECTED BY ADEQUATE MEANS.**

I agree to participate in the study conducted by the Human-Centered Computing Lab of the Freie Universität Berlin.

I understand that participation in this study is voluntary and I agree to immediately raise any concerns or areas of discomfort during the session with the study administrator.

I agree to be audio- and video-recorded for this test and understand that the recordings will be deleted after evaluation. I understand that all data gathered in this test will be anonymised. I am aware that results from these tests might be published.

Furthermore I agree to having pictures taken of the session, to be used as explanatory material in publications and for website news.

Please sign below to indicate that you have read and you understand the information on this form and that any questions you might have about the session have been answered.

The collected data will be controlled by
Freie Universität Berlin
Institut für Informatik
Arbeitsgruppe Human-Centered Computing (HCC)
Königin-Luise-Str. 24-26, 14195 Berlin

Name

Date

Signature

Thank you!

Human-Centered Computing
RESEARCH GROUP

Freie Universität Berlin

Pre-Assessment (Answer immediately)

How comfortable do you feel performing an ideation task with the software? Low ☐☐☐☐☐☐☐☐☐ High

Familiarisation

Task: You will now have 5 minutes to play around with Periscope to familiarise yourself with the interface. Please don't hesitate to ask any questions. You will see a number of stacks that offer various kinds of creativity prompts, principles and instances you may use for ideation purposes or to reflect your own ideas. There is no right way to use them, please feel free to use them as you see fit.

Before we continue, do you have any questions?

Pre-Assessment (Answer immediately)

How comfortable do you feel performing an ideation task with the software? Low ☐☐☐☐☐☐☐☐☐ High

Task

You will now be presented with a new session in Periscope that provides an environment for completing a particular task. The session includes, as before, various materials you may use to help you in creatively solving the challenges the task poses. You may now choose **one** of the following tasks:

- ☐ (A) Come up with **two** distinct and marketable innovations utilising **BIONIC RADAR** in the context of **INDUSTRIE 4.0**. Describe them in a way that someone else in your team may use them as sparks for their own ideation.
- ☐ (B) Explain the value of **one** distinct and marketable innovation utilising [**TECHNOLOGY**] in the context of [**YOUR AREA OF EXPERTISE**]. Describe it in a way that someone else in your team may use it as a sparks for their own ideation.
- ☐ ~~(C) Come up with a **task** that works well for **testing** and evaluating the effectiveness of **Periscope** for persons working on ideation task or creativity challenges in general.~~
- ☐ (D) Come up with **three** innovative **flavours of ice cream** for Jonny's Café in Berlin that appeals to Jonny's regular patrons who have seen it all. Jonny wants at least three ingredients, a story and plating concept for each flavour.

Checklist

Yes	No	
☐	☐	I have selected a task and understand it.
☐	☐	The software is set up in front of me on a notebook PC, I have a mouse and keyboard.
☐	☐	I am in a place where I can expect to work productively.

Before we continue, do you have any questions?

Please begin your selected task now.

Bionic Radar

Abstract technology description (contains no domain specific language)

The technology can perceive and analyse the movement of objects, such as humans, other living beings and things. By remembering the object's movement, it is capable of recognising it later on. Furthermore, the technology can compare the object's movement with the movement of other objects, and thereby conclude comprehensive movement patterns. These movement patterns enable the detection of objects with the same movement profile.

The described technology even functions in conditions of constrained visibility and complete darkness. Severe weather conditions and visual obstacles such as smoke, fog or fire pose no problem. Even thin biological substances such as human clothing or animal fur do not interfere with movement detection. However, metal and water remain opaque and cannot be penetrated. The technology is approximately hand-sized and can be used anywhere.

Metaphorical technology description

Like a bat, the technology can perceive the movement of objects, such as other humans, living beings or things. Once remembered, the technology can recognise these objects, just like a mother picking her child out of a crowd. By comparing the object's movement with the movement of other objects, the technology is capable of recognising comprehensive movement patterns, just like a good doctor who diagnoses a knee disease through observing and analysing a specific walking motion.

The described technology even functions in conditions of constrained visibility and complete darkness. Severe weather conditions and visual obstacles such as smoke, fog or fire pose no problem. Even thin biological substances such as human clothing or animal fur do not interfere with movement detection. However, metal and water remain opaque and cannot be penetrated. The technology is approximately hand-sized and can be used anywhere.

Bionic Radar

Detailed technology description (contains domain-specific language)

The technology (Bionic Radar) can perceive and analyse the movements of objects such as humans, other living beings and things. Therefore, it combines the functionality of two components: radar sensor and artificial intelligence.

First, the radar sensor emits electromagnetic waves, which are reflected by the respective object. The reflected signal then returns to the radar sensor and contains information about the object's characteristics and dynamics such as velocity, size, shape and distance.

Second, this object information is analysed through the means of artificial intelligence (AI). By storing the information in a database and comparing it with the movement data of other objects, the technology is able to recognise comprehensive movement patterns. Additionally, the AI component recognises objects that have previously been recorded, thus allowing for their exact identification.

The described technology even functions in conditions of constrained visibility and complete darkness. Severe weather conditions and visual obstacles such as smoke, fog or fire pose no problem. Even thin biological substances such as human clothing or animal fur do not interfere with movement detection. However, metal and water remain opaque and cannot be penetrated. The technology is approximately hand-sized and can be used anywhere.

Usefulness, satisfaction, and ease of use

Pertaining to the experience you just had in solving the task utilising Periscope, please make your evaluation now.

	Disagree	Agree	n/a
Usefulness			
It helps me be more productive.	☐☐☐☐☐☐☐		☐
It helps me be more effective.	☐☐☐☐☐☐☐		☐
It is useful.	☐☐☐☐☐☐☐		☐
It gives me more control.	☐☐☐☐☐☐☐		☐
It makes the things I want to easier to get done.	☐☐☐☐☐☐☐		☐
It meets my needs.	☐☐☐☐☐☐☐		☐
It does everything I would expect it to do.	☐☐☐☐☐☐☐		☐
Ease of use	Disagree	Agree	n/a
It is easy to use.	☐☐☐☐☐☐☐		☐
It is simple to use.	☐☐☐☐☐☐☐		☐
It is flexible.	☐☐☐☐☐☐☐		☐
Using it is effortless.	☐☐☐☐☐☐☐		☐
I can use it without written instructions.	☐☐☐☐☐☐☐		☐
Both occasional and regular users would like it.	☐☐☐☐☐☐☐		☐
I can recover from mistakes quickly and easily.	☐☐☐☐☐☐☐		☐
I can use it successfully every time.	☐☐☐☐☐☐☐		☐
Ease of learning	Disagree	Agree	n/a
I learned to use it quickly.	☐☐☐☐☐☐☐		☐
I easily remember how to use it.	☐☐☐☐☐☐☐		☐
It is easy to learn to use it.	☐☐☐☐☐☐☐		☐
I quickly became skillful with it.	☐☐☐☐☐☐☐		☐
Satisfaction	Disagree	Agree	n/a
I am satisfied with it.	☐☐☐☐☐☐☐		☐
I would recommend it to a friend.	☐☐☐☐☐☐☐		☐
It is fun to use.	☐☐☐☐☐☐☐		☐
It works the way I want it to work.	☐☐☐☐☐☐☐		☐
It is wonderful.	☐☐☐☐☐☐☐		☐
I feel I need to have it.	☐☐☐☐☐☐☐		☐
It is pleasant to use.	☐☐☐☐☐☐☐		☐

Please continue on the next page.

Usefulness, satisfaction, and ease of use (continued)

The most negative aspect(s):

The most positive aspect(s):

Interview

We will now conduct a 15 minute semi-structured interview. The goal of the interview is to discover what you think about Periscope, what benefits and what problems you see.

Thank you for your participation.

Do you have any questions?

Observer instructions 1

Introduction

PREPARATION

- ☐ 15” Notebook PC with external 3-button mouse with scroll-wheel.
- ☐ Confirm power supply.
- ☐ Confirm Do-Not-Disturb mode is engaged.
- ☐ Ask participant to turn the page
- ☐ A pen for the participant.
- ☐ Start recording screen, user-input, camera & microphone.
- ☐ Confirm recorder is running (Screenflick).

INSTRUCTIONS

- Read the script to the participant.
- Ask participant to take a seat at the notebook PC, sharing it with the observer.
- Answer all questions the participant has.
- Start timer when the participant is ready to begin.

SCRIPT

Today you will participate in the demonstration and evaluation of a prototype software called **Periscope**.

Please understand that we are **evaluating the software** and not you. Any and all confusion, misunderstanding and mistakes are the fault of the software and the design of this evaluation process. It is our goal to uncover as many issues with the software and the evaluation process as possible.

The evaluation will be conducted by a moderator/observer you can **ask questions** at any time. You are encouraged to **talk aloud** while interacting with the software, commenting on what you aim to do and which issues you encounter.

Your interactions with the software and commentary will be **recorded**. In particular the screen, keystrokes, mouse-movement, environment audio and your visible reactions.

Beginning each step of the evaluation you will receive written **instructions**. These instructions will also be read out to you. All instructions are contained in this manual. **Please do not read ahead. You will be asked to turn pages at appropriate moments.**

The evaluation will take approximately **45 minutes**:

5 min Software **demonstration**
5 min Casual **familiarisation**
15 min Solving a creativity **task** with the software
5 min Usability **questionnaire**
15 min Semi-structured **interview**

Observer instructions 2

Demonstration

PREPARATION

- ☐ Launch Periscope and load an empty session (New).
- ☐ Ask participant to turn the page.
- ☐ A pen for the participant.
- ☐ Confirm recorder is running (Screenflick).

INSTRUCTIONS

- Read the script to the participant while demonstrating the interface.
- Direct the participant to the assessment question.
- Answer all questions the participant has.
- Observe the time limit

SCRIPT

#0 Periskop hilft beim explorativen Nachdenken.

#1 Zum Beispiel, ganz allgemein bei Design Problemen oder bei der systematischen Suche nach Innovationen oder Lösungen die sich aus einer Situation heraus ergeben sollen, unter Anderem wenn man in einem Entwurf fest steckt oder sich im Kreis dreht.

#2 Periskop ist da um diesen Kreis aufzubrechen. Es bietet intuitive Mittel an, die das Denken spontan in Form bringen und nach Bedarf wilde/bizarre/absurde/ fremde Einflüsse abrufbar machen.

#3 Periskop bietet dafür Karten-Stapel mit Prinzipien, Beispielen, Ideen und Fragmenten an. Text, Bild, Ton, Video. Diese Stapel sind vorgefertigt, selbst gesammelt oder werden in einem Projekt unter allen geteilt.

#4 Periskop ist ein Werkzeug das einen beim Nachdenken begleitet.

#5 Und so schaut Periskop aus: Hier in der Mitte gibt es den Baum. Dort kann ich Karten anlegen. Zum Beispiel dazu was wir grade machen: "Periskop Schnellstart". (DEMO)

#6 Um Karten zu editieren muss ich nur Doppel-klicken. Doppelklick auf eine freie Stelle oder auf das Plus hier oben erzeugt eine neue Karte. Mit Alt-Plus kann ich duplizieren. Das X löscht Karten. Wenn ich will kann ich mit der Büroklammer ein bestimmtes Bild hinzufügen. (DEMO)

#7 Ich kann Karten auch beliebig hin und her schieben. Die Quadrate teilen sich automatisch und es gibt immer genug Platz (für etwa 16-tausend Karten). (DEMO)

#8 Ausser dem Baum gibt es noch die Stapel hier links. Stapel enthalten auch Karten, aber die können nur der Reihe nach durchgeblättert werden. Einfach scrollen. (DEMO)

#9 Wo der Baum Übersicht gibt, kümmern sich Stapel darum dass man immer wieder zufällig an Inspirationen vorbeikommt. (DEMO)

Observer instructions 3

Familiarisation

PREPARATION

- ☐ Launch Periscope and load the "Quickstart.0.5.x.json" session.
- ☐ Ask participant to turn the page
- ☐ A pen for the participant.
- ☐ Confirm recorder is running (Screenflick).

INSTRUCTIONS

- Offer printed instructions to participant.
- Direct the participant to the assessment question.
- Read the script to the participant.
- Answer questions the participant has.
- Observe the participant and take note of all things not captured by the recording. Pay particular attention how the participant interprets the UI features and how they (expect to) utilise them.
- Encourage the participant to **talk aloud**.
- Observe the time limit

SCRIPT

You will now have 5 minutes to play around with Periscope to familiarise yourself with the interface. Please don't hesitate to ask any questions. You will see a number of stacks that offer various kinds of creativity prompts, principles and instances you may use for ideation purposes or to reflect your own ideas. There is no right way to use them, please feel free to use them as you see fit.

Observer instructions 2

Demonstration

SCRIPT (CONT.)

#10 Stapel wirken auch wie eine Zwischenablage. Ich kann einfach Karten die grade nicht passen aussortieren. Z.B, auf Nummer 6. Ich kann Stapel mit 1-6 auswählen, dann werden sie größer und ich sehe mehr. (DEMO)

#11 Und ich kann so auch markierte karten mit Alt+X direkt auf den Stapel werfen. (DEMO)

#12 Jetzt, wenn ich meine ersten Notizen gemacht habe kann ich fröhlich zwischen allen Karten herum-navigieren. Mit den Pfeiltasten (DEMO). F zoomt die Kamera so dass alle ausgewählten karten maximal groß sichtbar sind (DEMO) und Alt-F setzt die Kamera zurück. (DEMO)

#13 Um Periskop effektiv zu nutzen muss man sich ein paar Muster überlegen wie man Karten organisiert. Man kann sie einfach in Quadrate werfen aber das kann evtl. verwirrend werden.

#14 Eine gute Idee ist es, in einem Quadrat immer eine Ecke plakativ mit einem Schlüsselmotiv oder Thema zu versehen und die übrigen Quadrate für Notizen, Highlights und einen Sammelordner zu verwenden. (DEMO)

#15 (Auf Fragen eingehen)

#16 Du kannst jetzt gleich mal 5 Minuten selber probieren. Fragen?

Observer instructions 5

Usefulness, satisfaction, and ease of use

PREPARATION

- ☐ Ask participant to turn the page
- ☐ A pen for the participant.
- ☐ Confirm recorder is running (Tascam).
- ☐ Confirm "all boxes are checked".

INSTRUCTIONS

- Read the script to the participant.
- Answer questions the participant has.
- Observe the time limit.

SCRIPT

Pertaining to the experience you just had in solving the task utilising Periscope, please make your evaluation now.

Observer instructions 4

Task

PREPARATION

- ☐ Ask participant to turn the page
- ☐ A pen for the participant.
- ☐ Confirm recorder is running (Screenflick).
- ☐ Launch Periscope and load the "Bionic Radar.json" or "Ice Cream.json" session based on the participant's choice.
- ☐ Confirm "all boxes are checked".

INSTRUCTIONS

- Offer printed instructions to participant.
- Direct the participant to the assessment question.
- Read the script to the participant.
- Answer questions the participant has.
- Observe the participant and take note of all things not captured by the recording. Pay particular attention how the participant interprets the UI features and how they (expect to) utilise them.
- Encourage the participant to **talk aloud**.
- Observe the time limit
- **When done**: Stop the recording.

SCRIPT

You will be presented with a new session in Periscope that provides an environment for completing a particular task. The session includes, as before, various materials you may use to help you in creatively solving the challenges the task poses. You may choose one of the following tasks:

(A) Come up with two distinct and marketable innovations utilising BIONIC RADAR in the context of INDUSTRIE 4.0. Describe them in a way that someone else in your team may use them as sparks for their own ideation.

(B) Explain the value of one distinct and marketable innovation utilising BIONIC RADAR in the context of [YOUR AREA OF EXPERTISE]. Describe it in a way that someone else in your team may use it as sparks for their own ideation.

(C) Come up with a task that works well for testing and evaluating the effectiveness of Periscope for persons working on ideation task or creativity challenges in general.

(D) Come up with three innovative flavours of ice cream for Jonny's Café in Berlin that appeals to Jonny's regular patrons who have seen it all. Jonny wants at least three ingredients, a story and plating concept for each flavour.

Once you have selected your task, you have 15 minutes to work on it. Please note, the intention is to give you opportunity to work with Periscope in a directed and focused manner.

Observer Notes *

Observer instructions 6

Interview

PREPARATION

- ☐ Ask participant to turn the page.
- ☐ A pen for the participant.
- ☐ Confirm recorder is running (Tascam).

INSTRUCTIONS

- Read the script to the participant.
- Answer questions.
- Ask questions.
- Observe the time limit.
- Observe the participant and take note of all things not captured by the recording. Pay particular attention how the participant interprets the UI features and how they (expect to) utilise them.

SCRIPT

We will now conduct a 15 minute semi-structured interview. The goal of the interview is to discover what you think about Periscope, what benefits and what problems you see.

Q: What was your **first impression** of the application was and how it changed over the course of the session today?

Q: What **didn't** you **like** about the application?

Q: What did you **like** about the application?

Q: Did you see the application **help** in solving any problem relating to the task you were given today?

Q: How was the application able to facilitate your **intentions**.

Q: How did the application as a whole or elements of it **influence** the ideation task?

Q: How the **test situation influence** the outcome of the ideation task?

Thank you.