Towards a Computational Model of Abstraction in Design Reasoning

Ryan Bruggeman, Estefania Ciliotta Chehade, Paolo Ciuccarelli
(bruggeman.r, e.ciliottachehade, p.ciccarelli@northeastern.ed)
Center for Design, Northeastern University, 11 Leon Street
Boston, MA 02115 USA

Tucker J. Marion (t.marion@northeastern.edu)
Entrepreneurship & Innovation, 370 Huntington Avenue
Boston, MA 02115 USA

Abstract

This paper seeks to understand designers' abstraction in ill-structured problem-solving. We utilize a protocol study with expert designers to empirically analyze the abstraction process in the latent need problem setting. A logic-based abstraction schema is found to model the process the designers employed. The study reveals how designers utilize this schema, detailing, developing, and evaluating solutions for ill-structured problems. It highlights the recursive nature of abstraction and raises questions about the termination of the process in ill-structured domains. We conclude by proposing a computational model to further evaluate abstraction in complex problem-solving scenarios.

Keywords: Abstraction; Reasoning; Ill-Structured Problems; Computational Model; Design; Protocol

Design & Ill-Structured Problems

Studies of designers have been central to understanding the problem-solving capabilities needed for ill-structured problems (Hay, Cash, & McKilligan, 2020; Simon, 1977). This is important because, unlike well-structured problems with clearly defined solutions and modes of operations, such as simple arithmetic, ill-structured problems do not have the luxury of being clearly defined and knowable at the outset, leading to open-ended outcomes. An ill-structured problem domain is one where (a) the problem definition and (b) the mode of operation for working within that space are not known (Simon, 1977). This definition can be applied to a group of reasoners who are not knowledgeable about the problem definition or mode of operation and, as such, require abstraction.

Such circumstances give rise to a search process (Hay et al., 2017; Simon, 1977), where the reasoner accumulates information about the open-ended solution using an abstract knowledge structure containing both declarative knowledge about the problem and procedural knowledge (Ball, Ormerod, & Morley, 2004). The reasoner applies a schema, not to elicit a problem solution per se, but a possible combination of information to allow the solution to be known (Simon, 1977). Simon states that when developing a schema for ill-structured problems, the reasoner must create a plan that moves them beyond the original problem space into an abstract space to accumulate the best representation of information. This is corroborated by Gentner & Hoyos (2017), who define abstraction as "decreasing the specificity (and thereby increasing the scope) of a concept." Abstraction develops a many-to-one mapping from the original problem space to the abstract space to relate and categorize the key characteristics of the problem.

The effort to perform this search involves a comprehensive application of reasoning through multiple inference steps. Mitchell (2023) describes the composing multiple steps of inference as abstraction acting as an umbrella term that is not limited to any one form of inference, i.e., inductive, deductive, abductive, analogical, and case-based. Studies of abstract reasoning have been performed on several types of agents, namely children and primates (Gentner & Hoyos, 2017; Starkey, Spelke, & Gelman, 1990; Sampson et al., 2018), and in contexts, namely language (Bransford & Franks, 1971; Tomasello, 2001), mathematics (Koedinger & Anderson, 1990), and physics (Shin & Gerstenberg, 2023). Human infants, for example, who are unaware of rigorous mathematics, demonstrate the ability to detect numerical information (Starkey et al., 1990). Through several experiments, the researchers found that infants expressed an ability to abstract and form equivalence and non-equivalence relations between numbers. Another group demonstrated that rhesus monkeys in reverse-reward problems could comprehend non-perceptual features, infer them from one specific case, and use them to override the natural preference to select the superior option (Sampson et al., 2018).

These findings exemplify that abstraction is called upon when a reasoner is (a) confronted with an unknown domain and (b) how to operate in that domain. Yet, the reasoner must execute some process to satisfy the problem. Gentner & Hoyos (2017) suggest that we focus on the process the reasoner applies to further understand how a reasoner acquires and performs such abstractions. This design case is of particular interest because the entirety of the discipline is made to tackle ill-defined problems with unknown modes of operation (Buchanan, 1992). We can think of the design of experiments, systems, applications, and products as outcomes that result from a process to accumulate information, such as dimensions, structural limitations, and form and functional preference, to resolve the problem. The expert designer is of particular interest for two purposes: as a reasoner, unlike the infant or rhesus monkey who forms an abstraction where communication is limited, the expert designer can verbally and visually communicate their abstract reasoning (Koedinger & Anderson, 1990); and context; the development of the inference procedure is performed in a complex setting where the end state is unknown and therefore provides insight into the reasoning necessary to perform meaningful abstraction.

Our Work

One such ill-structured problem that calls for abstraction by designers is latent user need finding (Carlgren, 2013). When looking at online user reviews, designers must elicit information that will be meaningful for the design or redesign of a product. Online user reviews provide such information about the use and contentment of the user using that product, implying that if that area of nuisance is corrected, the product will improve. Such information is relayed explicitly by the user as evidence of love or hate for a feature of a product. Other information can be implicit, where the user has purposefully omitted information, deeming it irrelevant to the review, or they had been vague (Carlgren, 2013). This would suggest the existence of latent user needs. Latent user needs are demands, problems, or goals a user may have for a product or service but are, until then, unforeseen by the user. The user does not know that those "asks" exist of the product and are left inactive and unexpressed as a desire by the user. Therefore, a latent user need is a need that has the potential to arise for a user but is, until then, unforeseen by the user. Ill-structuredness becomes apparent because one can look at the language syntax and propose the need for that user group, but how do we know it is a latent need and not an explicit one?

In this paper, we study designers' abstract reasoning for ill-structured problems. We conduct a protocol study with expert designers to empirically test and elicit the representations developed when latent user need finding. We found that the results of the visual and verbal protocols are predicted by a proposed logic model by (Bruggeman, Ciliotta Chehade, & Ciuccarelli, 2023), which symbolizes the relational abstractions used by designers in ill-structured problems. We discuss these results as a foundation for a cognitive computational model to further study designer abstraction.

Abstract Reasoning in Design

Designers apply abstraction when the problem exhibits multiple, interacting, complex behaviours, complex boundaries and interactions between components, and many-to-many relationships between behaviours and forms (Hoover, Rinderle, & Finger, 1991). When confronting such a problem, the designer will develop a model to predict the potential solution's behaviour and help make refinements while inferring the solution. During abstraction, designers do not maintain the description of the solution at a uniform level of detail throughout the process due to complexity but instead focus on certain parts of the problem in detail and ignore other parts irregularly, making refinements when new information arises or an impasse towards the solution emerges.

Gero (2000) proposes that the goal of the process is to transform some function into a description so that the solution being described can produce those functions. He uses the example of windows, where some functions include the provision of daylight, control of ventilation, and access to a view. When the complexity of the problem increases, the elicitation of functions and their translation into descriptions becomes increasingly difficult, requiring the designer to develop and search the space of the problem more comprehensively.

Designer's abstract models of the problem space have additionally been discussed as mental models (Johnson-Laird, 2010; Hay et al., 2017). The designer's reasoning depends on some tacit form of mental logic to constructively develop and search within the space. Goel (1995) describes a designer's abstraction via vertical and horizonal transformations. *Horizonal transformations* develops the problem from one stage to a slightly different stage, "widening the space," whilst *vertical transformations* transition the problem to a more detailed version of itself, "deepening the space." These two transformations are contrasted within an *abstraction hierarchy*, where the horizontal transformation is the development of the type of information being considered and the vertical deals with the information's generality or detail. Similar schemas have been used to describe abstraction in mathematics, wherein horizontal abstraction deploys diagrams and symbols to represent the essential underlying relationships and identify irrelevant aspects of the problem to ignore, whilst vertical abstraction is a process that leads to the formation of one or more new mental objects at a higher level of generality in the form of hierarchy (Mitchelmore and White, 2012).

A Logic-Based Schema of Design Abstraction

A logic-based schema enables a representation of abstract reasoning without limiting the possible type of inference being made within (Hummel & Doumas, 2023). In the case of design, the abductive-deductive formulation has been the predominant model to detail designers' abstract reasoning (Koskela, Paavola, & Kroll, 2018; March, 1972). Based on the work of C.S. Pierce (1933), the creation of a solution does not rely just on a deduction from previous facts but requires a novel form of reasoning to arrive at a desirable outcome. Abduction models are an explanatory hypothesis that the reasoner can further rework their abstraction. Empirically, Cramer-Peterson, Christensen, & Ahmed-Kristensen (2019) found through 218 idea design activities that abductive-deductive problem development dominates designer ideation through a cycle of analysis-synthesis. Abduction proposes frames or perspectives for addressing the problem, while deduction, in turn, explores how such a frame can address the problem.

Empirical developments suggest using an abductive-deductive process is predominant in design to perform a vertical and horizontal transformation to develop and search the problem space (Kroll, Le Masson, & Weil, 2023). To situate these findings, Bruggeman et al. developed an abductive-deductive logic-based schema to model how an ill-structured problem space is searched and developed when navigated by the designer. In the schema, the characteristics of the ill-structured domain are captured as follows: the problem space is defined using symbolic variables, whilst the reasoner knowledge is represented by k, which indicates epistemic operations performed by the designer.

Table 1: Inference rules used to develop ill-structured search space.

Inference Rule	*Formalism*
Induction:	$Th \cup F \vdash_\sigma A$
Deduction:	$A \cup F \vdash_\sigma Th$
Abduction:	$A \cup Th \vdash_\sigma F$

Three variables and inference rules describe the designer's development of the problem space (Table 1). Known axioms A are ontological truths or rules; facts F are established by the reasoner; Th is a set of theorems/hypotheses that can be put forth. In deduction, we may have A: Newton's laws of gravity; F: observation of planetary motion; Th: the theory of universal gravitation. This would be done using an inference rule $\vdash_\sigma$ such as modes ponens.

Bruggeman et al. demonstrated that deduction was not strong enough to develop the problem space when that space is ill-structured. Via deduction alone, there are no logical guarantees that the Th it provides offers a complete representation of all information. Instead, it might be a snapshot or component of the possible solution to the problem. To expand the deductive approach, they conjoin Th with an epistemic operator $Th(k)$, and when unified with A via abduction, model a new fact $F(k)$,

$$\text{(Eq. 1)} \quad A \cup Th(k) \vDash F(k)$$

Furthermore, if there is a set $Th(x)$, there is a delineation that can be made between the reasoner's knowledge $k(x)$ and $F(x)$. The two can be unified to infer the theory in terms of their knowledge about x: $Th(k(x))$. As such, by abduction $Th(k(x))$ can be unified with A, to abduce a F integrated with knowledge about that fact: $F(k(x))$. This allows the reasoner to develop a detailed account about the $F(x)$'s of the problem based upon one's knowledge about x and A.

The schema is made more robust depending on the reasoners k. When they have met the limits of x, in what it captures of the problem space, they can combine $F(k(x))$ with a new set of $Th(y)$ through k. This enables the modelling of a multivariable representation $F(k(x, y))$, enabling horizontal information development (Eq. 2 and 3).

$$\text{(Eq. 2)} \quad (k(y) \cup F(y)) \cup F(k(x)) \vdash Th(k(x, y))$$
$$\text{(Eq. 3)} \quad A \cup Th(k(x, y)) \vDash F(k(x, y))$$

Bruggeman et al. provide an argument for the soundness and completeness of this schema, but empirically, it is not validated. Logically, soundness holds when k is modelled with A on the condition of being conjoined with the deduced Th; completeness holds when k is what unifies the multiple dimensions of the problem. This suggests that to validate the proposed schema of abstraction for ill-structured problems empirically, we need to address the use of k when conducting the search and development of the problem. In other words, the model defines how space develops, but this depends on how the k searches within the space.

Designer Protocol Study

We performed a protocol study with expert designers to empirically validate k in the designer's abstract reasoning. A protocol enables researchers to elicit verbal and visual reasoning from participants conducting tasks (Ericsson and Simon, 1993). This is ideal for understanding the designer's reasoning in an unrestrictive manner with little interference on the researcher's part.

The logic-based schema of Bruggeman et al. was derived in the context of latent user need-finding, and as the researcher's state, designers are best equipped to identify latent user needs. The protocol study we have conducted is interested in the use of k as it is suggested to perform by the schema, not in validating the outcome itself — this is because ill-structured problems can have many representations, as is the case with latent user needs, and cannot be known to be "best" until further user research is conducted. The validity of the outcome is self-contained by the model's logic, and validating the use of k will, in part, validate the outcomes it produces. As such, the protocol aims to identify k as it (1) vertically details the variables in the problem and (2) horizontally combines information to create a more detailed representation of the solution.

Experiment

The protocol was conducted with 10 expert designers with 5+ years of user experience research as part of their design practice. Nielson & Landaur (1993) found that the optimal sample size for qualitative testing was 5 participants. As such, we divided the participants such that 5 performed the exercise on Protocol 1: a set of 10 reviews/annotations for a distinct shoe, while a control group of 5 performed the same exercise on Protocol 2: another 10 reviews/annotations for a distinct shoe. None had any prior experience with latent user need elicitation. Each protocol happened individually over a 2-hour period, where the participant's screen and video were recorded and transcribed through Microsoft Teams. The protocol took place on Miro, an online whiteboard software that allows for live visualization and alteration of information. The objective of the think-aloud was for designers to *elicit the latent user needs from online user reviews for the redesign of a shoe.* We pulled reviews from 2500 online products and found that the average number of reviews on a product was 10. The average length of a review ranged from 3-5 sentences. We created two separate Miro boards, each for a different shoe, where the participants were presented with 10 user reviews ranging from 3-5 sentences.

With each review, the designer received an annotated version of each review. Han et al. (2023) developed an annotation model for sentiment analysis that can automatically partition user reviews at a sentence level into terms representing the *category* (CAT), *aspect* (ASP), *opinion* (OP), and *sentiment* (SEN) of the review. The ASP is the objective target of the sentence, usually a noun or verb, ex. shoelace, run, house. OP is the user's subjective statement in the sentence, such as 'I like…' or 'They felt…'. SEN tags the positive, negative, or neutral sentiment of the OP term. CAT is the ontological category the ASP belongs to; ex. *leather* is part of the *Appearance#Material* category. A sentence would be annotated as follows: "*The leather is a nice look.*"; {ASP: "*leather*"; CAT: *Appearance#Material*; OP: "*looks good*"; SEN: *Positive*}.

The purpose of the annotated reviews was to provide workable parts of the review to streamline reasoning, though they were not restricted from working directly from the review itself. The ASP and OP represent *F* as the user establishes contingent observations. SEN and CAT represent *A*, as these necessarily hold ontologically. We observe their use to construct *Th* by the designer's use of *k*.

Results

We used thematic analysis to develop the mapping between the designer's protocols and the logical schema (Saldaña, 2009). Each protocol produced a transcript of the designer's verbal thinking and screen recording videos of visual reasoning — some designers leaned towards one or the other form of thinking aloud or utilized both evenly. The coding names and definitions to identify *Th* development and *k* in the data are drawn from previous literature in design cognition, specifically, the actions found to be most common in design reasoning (Hay et al., 2017).

Each researcher went through an initial coding phase where they coded only explicit verbal or visual actions taken by the designers. This required identifying verbal or visual actions that supported the designer's reasoning about the problem. For example, a designer stated, *"I'm ignoring a lot of the positive like what I deem is just kind of almost like flowery statements because to me, and this is the first goal or problem I have is there's some performance and usage issues and those might need to be what I focus on and later I'll acknowledge that like look or design was fine."* This epistemic action was encoded as a *rule application,* where the designer would form a rule to delineate between symbols to build their representation. When such actions were observed, the time would be marked, and we would encode how the designer used ASP, OP, CAT, and SEN. For the above, *instantiation* followed *rule application*, where the designer saves symbols as part of the problem representation, i.e., SEN ∪ OP ⊦ SEN(OP):

Negative ∪ *"very loud as your walking as they bend"* ⊦ *Negative*(*"very loud as your walking as they bend*").

To conclude the analysis, the researchers triangulated their codes to categorize themes and subthemes. We found through frequency analysis (Table 2) that the designer's abstraction occurs in three phases: *Detail, Develop*, and *Evaluate*. Below

Table 2: The percentage with which the 10 designers employed the same action. The names and definitions of the actions are adapted from Hay et al. (2017)

Action	*Frequency* (-/10)
Phase 1: Detail	
Enquire – instantiate symbols to address a need for information.	1.0
Generalize – associate a symbol to a supra-symbol.	0.2
Goal Definition – define goals/subgoals.	0.1
Inference – hypothesize new symbol relationship.	0.8
Integrate – further specify current solution state.	0.8
Instantiate – save new symbol as part of problem representation.	1.0
Represent – create external representation.	1.0
Rule Application – develop/use established reasoning rule (arithmetic, logical, assertion).	0.7
Phase 2: Develop	
Generate – create connection between information.	1.0
Modify – alter connections between information.	0.9
Speculation – produce partial solution or specification.	1.0
Case Based – compare information to previous experience.	0.7
Analogy – use information about known semantic concepts to understand newly presented concepts in the information.	0.5
Inference – logical judgement based on pre-existing information.	0.7
Affinity – mapped visual affinities between created information to determine relationships.	0.5
Contradiction – judge validity of information based on compatibility with other information.	0.2
Phase 3: Evaluate	
Compare – determine compatibility of proposal to constraints.	1.0
Calculate – infer new information by combining existing information.	1.0
Patch – add/combine information without making less abstract.	1.0
Evaluate – assess information.	0.7
Simulate – Represent information at proper level of abstraction in order to relate it.	1.0
Accept – Add new information to solution state.	1.0
Reject – Determine information unsatisfactory.	0.8
Refine – Make information more specific.	0.9

we discuss the three phases as they were found in the protocols by outlining the patterns of operators used and their mapping to the logic schema.

Detail The first phase of the designer's abstraction saw a vertical detailing of the user information. Each designer would begin by processing the reviews to build a representation of the problem. This required reading and visualizing the information they deemed most relevant to the problem: *enquire.* Each designer then built a *representation* of the enquired information by classifying clusters $k[a_i, b_i, c_i, \ldots]$. Each cluster k represented information about the user, such as sizing/fit issues with the shoe. To further develop the representations, designers would follow a recursive application of *instantiate–inference–integrate.* Logically, the pattern followed the map:

$$\text{(Eq. 4)} \quad k(a_i) \cup \mathrm{SEN} \vdash \mathrm{SEN}(a_i)$$
$$\text{(Eq. 5)} \quad k(a_i) \cup \mathrm{OP}(a_i) \vdash \mathrm{SEN}(\mathrm{OP}(k(a_i)))$$
$$\text{(Eq. 6)} \quad \mathrm{SEN}(a_i) \cup \mathrm{SEN}(\mathrm{OP}(k(a_i))) \vDash \mathrm{OP}(k(a_i)).$$

Of course, this is only one such example. The designers would utilize this schema to detail their $k[a_i, b_i, c_i, \ldots]$ using ASP, CAT, OP, and SEN from each respective review. The goal for each designer was to relate parts found in the review sentences, i.e., ASP and OP, to their knowledge k. As seen in the above example, the logic was applied to relate the ASP and OP to k through their axiomatic representations, i.e., CAT and SEN.

Johnson-Laird's (1983) procedure of syllogistic reasoning describes the tableau the reasoner first details as organizing elements that stand for members of sets. The designer would represent members of sets through their relation to k. This is substantiated by how the designer built their model of the annotation variables, keeping track of the relations of the terms in each review set to a specific cluster of k. This stage is analogous to vertical abstraction, forming a hierarchical generality in the information.

Develop In the second phase, the designers sought to connect their detailed representations of $k[a_i, b_i, c_i, \ldots]$ and their respective annotations. The designers would begin by *generating* a connection between a detailed representation of $k(a_i)$ and $k(b_i)$, for example, the $\mathrm{OP}(k(a_i))$ and $\mathrm{ASP}(k(a_i))$ to $\mathrm{OP}(k(b_i))$ and $\mathrm{ASP}(k(b_i))$. To validate the generated connection, the designers would apply some reasoning rules (*case based, analogy, inference, affinity, contradiction*) to modify the information. *Generation*:

$$\text{(Eq. 7)} \quad \mathrm{OP}(k(a_i)) \cup \mathrm{ASP}(k(b_i)) \vdash \mathrm{OP\text{-}ASP}(k(a_i, b_i))$$

The temporary structure between a_i and b_j is then tested via one of the *rules,* creating a *specification*. The newly specified theory would then *modify* the model (Eq. 8).

$$\text{(Eq. 8)} \quad \mathrm{CAT}(a_i, b_j) \cup \mathrm{OP\text{-}ASP}(k(a_i, b_j)) \vDash \mathrm{CAT}(\mathrm{OP\text{-}ASP}(k(a_i, b_j)))$$

This is repeated until a model is developed that has integrated a representation with all variables and dimensions. During this phase, if knowledge of certain variables were seen to contradict one another or were false under the rules, they would be rejected or partitioned. The Develop phase adds second premises to the first premises, considering the different ways this can be done (Johnson-Laird, 1983). The horizontal transformation is embodied in this stage of the designer's abstraction, which establishes underlying relationships that connect different information found portent to the problem. This is notable in the rules the designer uses to modify facts and combine premises $[a, b, c\ldots]$ to restructure their model.

Evaluate Develop saw the designer generate a robust, multidimensional representation of the variables in a unified model. The designer would then distinguish between parts of the model for each latent and explicit user need. The designer would *compare* parts of the model, *calculate* the differences between different representations, and create a *patch*, adding or combining the information they found in the calculation to the latent need. The designer would conclude by *simulating* the model at a proper level of abstraction to relate it to the problem, *evaluate* the outcome by voicing their thoughts about it, and then decide to *accept, refine*, or *reject* the resulting simulation. If the designer is determined to refine, they will repeat the compare-calculate-patch chain.

The Evaluation phase frames a conclusion to express the relation, if any, between the end terms that hold in all the models of the premises (Johnson-Laird, 1983). This is elucidated by the participants' use of the *compare-calculate-patch* chaining, which is used to develop the latent user needs. Framing the conclusion first involved simulating and then accepting, rejecting, or refining the simulation through an additional compare-calculate-patch chain until the premises held.

A Computational Model of Design Abstraction

The logical reasoning schema is abstract and procedural, "where knowledge elements do not correspond to any particular situation or set of objects, but to large categories of situations, and they prescribe an action" (Koedinger & Anderson, 1990). In the ill-structured problem domain, using k and the inference patterns exhibited by the designer is pertinent to operating in an unknown domain. k was used to substantiate the conceptual clusters of the user review data, whilst inferential procedures (induction, deduction, and abduction) enabled the development of those clusters in relation to one another. Pierce (1933) characterizes this as 'hypostatic abstraction': the taking of a precept that has propositional form in a judgment, "and in conceiving this fact to consist in the relation between the subject of that judgment and another subject." He uses the example of transforming the proposition "honey is sweet," X is Y, to "honey has sweetness," X has the property of being Y. This is the act of turning predicates that are signs we think through,

Phase 1: Detail	Phase 2: Develop	Phase 3: Evaluate
k(*a*): *Materials* = “plastic type of material”, “after wearing these shoes twice, the upper learther ripped”; “was sturdily constructed”, “slightly different the stitching is different that sound feeling is different”, “poke right through the cheap material by the toe” *k*(*b*): *Product Perception* = “I am dissapointed in the quality and had higher expectations”, “do not believe that these sneakers are genuine leather”, “just didn’t feel like the classic running shoes that I had before”, “I have always ordered this shoe”, “looks and feels terrific for everyday wear”, “sizing run smalls”, “not comfortable” *k*(*c*): *Consequences* = “I sent them back and order me a pair a Reebok Classic running instead”, “Looking elsewhere or for another brand”, “In the last few years something has changed”	“Usually what I also start doing my *affinity map* already before highlighting too much. I go in sections so I'll try and and do it simultaneously or side by side.” “I had purchase issues in my last... so slightly different stitching... so here we have material.” “I just didn't feel like a classic running shoes.” “This is more product perception and this is the consequence you're exactly like my last.” “So you can already see that this is more materials not comfortable, that's a negative product perception.” “Now it's more material quality, product cost and then the perceived value, which is an interesting slight pattern that we see already ”	Material quality is percieved low. ↓ Product costs is percieved high. ↓ Percieved value declines. *Latent Need*: Being familiar with the same product over time raises certain expectations in material quality and price perception. Even when the product changes over time, the same level of quality is expected.

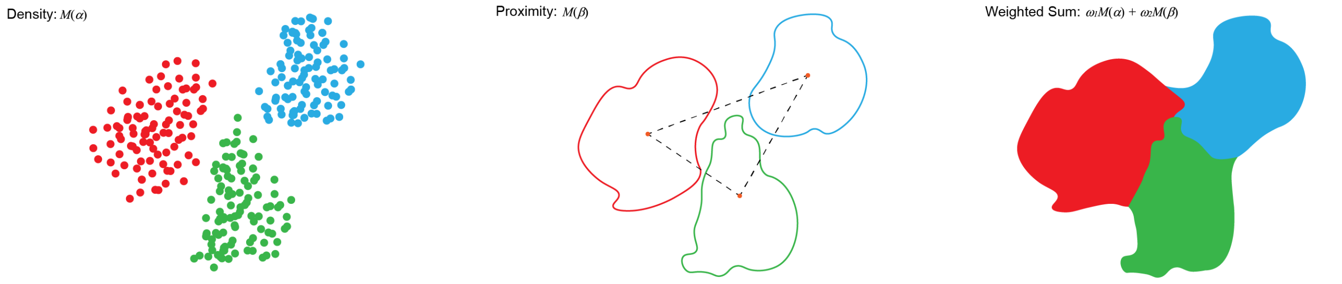

Figure 1: The three phases of abstraction for a participant designer illustrate the proposed regression model below. *Phase 1: Detail*, the designer emphasized user OPs. They listed the user opinions under three main *k* clusters. An example *density* map of the clustered opinions respective to each list is below. *Phase 2: Develop*, provides a snippet of the designer applying affinity mapping to establish connections between the three clusters. Below is the *proximity* that can be calculated to each cluster’s centroid using cosine similarity or Euclidian distance, for example. *Phase 3: Evaluate*, exemplifies the designer's refined latent user need. Below illustrates the *weighted sum* that can be taken of the *density* + *proximity* evaluation to map a score to the outputted statement.

to being subjects thought of. The process thus repeats, allowing us to form propositions of second and third degree, etc. By nature, recursive, this results in what Pierce calls a ‘continuous predicate.’ Given the nature of the ill-structured problem, we do not know the outcome. So, the final output during abstraction is continuous, wherein other relations can be further made if facts or axioms become known. Although designers followed the logic-based schema, they rarely arrived at the same latent need (Protocol 1: 3/5 were the same; Protocol 2: 2/5 were the same), let alone the same level of detail.

We propose developing a computational model to further evaluate abstraction in ill-structured problems (Figure 1). We can conclude from our study that there is a relationship between a model M that contains k, which the designer established, and some targeted phenomenon P. Furthermore, the logic schema demonstrates there is a relation Λ that accurately models the relation between M and P, as in the logic-based schema, such that,

$$(\dagger) \qquad \Lambda(M, P) \leftrightarrow \Omega$$

In a discrete setting Ω, if M exhibits an ability to predict P, then there is a biconditional such that Λ is the relation map constituting M’s prediction of P (Bringsjord, Giancola, & Govindarajulu, 2023). Furthermore, if Ω is a setting where we can check that M successfully predicted P, then successful prediction must be correlated with Λ.

We seek to map the relationship between M, which the model developed during the Detail, Develop and Evaluate phase, to P, which is the proposed solution state, a latent need statement in our circumstance. Λ represents the characteristics that accurately model P. We can split M into Detail: α; and Develop: β. In Detail, designers established clusters of OP and ASP related to k. We can use $M(\alpha)$ to evaluate the *density* of each $k[a_i, b_i, c_i, \ldots]$ to determine the detail (i.e., the presence of OP and ASP in each cluster). During Develop, designers established connections between each cluster; as such, we propose using a *proximity* metric for $M(\beta)$ to evaluate the similarity between each cluster. Furthermore, each designer weighted phases differently in terms of importance, as such weights ω_1 and ω_2 can be attributed to $M(\alpha, \beta)$, respectively. Finally, for Evaluate, we can take a weighted sum of our clusters to establish $\Lambda(M, P)$:

$$(\ddagger) \qquad P = \omega_1 M(\alpha) + \omega_2 M(\beta)$$

Conclusion

This study focuses on expert designers who can communicate their abstract reasoning verbally and visually. We conducted a protocol study with expert designers to test and elicit representations developed from reasoning about an ill-structured problem. The results align with the proposed logic-based schema, symbolizing relational abstractions used by designers. However, we found that the logic-based schema only accounted for the development and search of the ill-structured problem space. It remained unclear why the abstraction process was terminated during the Evaluation phase. To conclude, we propose a computational model to evaluate designer abstraction in a future study.

Acknowledgments

This material is based upon work supported by the National Science Foundation under the Engineering Design and System Engineering (EDSE) Grant No. 2050052. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. Thank you to Mohsen Moghaddam and Lu Wang.

References

Ball, L. J., Ormerod, T. C. & Morley, N. J. (2004) Spontaneous analogising in engineering design: a comparative analysis of experts and novices. *Design Studies, 25*(5), 495–508. doi:10.1016/j.destud.2004.05.-004.

Bransford, J. D., & Franks, J. J. (1971). The abstraction of linguistic ideas. *Cognitive Psychology, 2*(4), 331–350. https://doi.org/10.1016/0010-0285(71)90019-3.

Bringsjord, S., Giancola, M., & Govindarajulu, N. S. (2023). Logic-Based Modeling of Cognition. In R. Sun (Ed.), *The Cambridge Handbook of Computational Cognitive Sciences*, 173–209. chapter, Cambridge: Cambridge University Press.

Bruggeman, R., Ciliotta Chehade, E., & Ciuccarelli, P. (2023). Expanding User Need Finding Through Abductive Reasoning. *Proceedings of the Design Society, 3*, 1745–1754. doi:10.1017/pds.2023.175.

Buchanan, R. (1992). Wicked Problems in Design Thinking. *Design Issues, 8*(2), 5-21. http://www.jstor.org/stable/-1511637.

Carlgren, L. (2013). Identifying latent needs: Towards a competence perspective on attractive quality creation, *Total Quality Management & Business Excellence, 24*(11-12), 1347–1363. https://doi.org/10.1080/14783363.2013.-776762.

Cramer-Petersen, C. L., Christensen, B. T., & Ahmed-Kristensen, S. (2019). Empirically analysing design reasoning patterns: Abductive-deductive reasoning patterns dominate design idea generation. *Design Studies, 60*, 39–70. https://doi.org/10.1016/j.destud.2018.10.001.

Ericsson, K. A., & Simon, H. A. (1993). *Protocol analysis.* MIT Press. https://doi.org/10.7551/mitpress/5657.001.-0001.

Gentner, D., & Hoyos, C. (2017). Analogy and abstraction. *Topics in Cognitive Science, 9*(3), 672–693. https://-doi.org/10.1111/tops.12278.

Gero, J. S. (2000). Computational models of innovative and Creative Design Processes. *Technological Forecasting and Social Change, 64*(2–3), 183–196. https://doi.org/-10.1016/s0040-1625(99)00105-5.

Goel, V. (1995). *Sketches of thought.* MIT Press.

Han, Y., Bruggeman, R., Peper, J., Ciliotta Chehade, E., Marion, T., Ciuccarelli, P., & Moghaddam, M. (2023). Extracting Latent Needs From Online Reviews Through Deep Learning Based Language Model. *Proceedings of the Design Society, 3*, 1855–1864. doi:10.1017/pds.2023.186.

Hay, L., Duffy, A. H. B., McTeague, C., Pidgeon, L. M., Vuletic, T., & Grealy, M. (2017). A systematic review of protocol studies on conceptual design cognition: Design as search and exploration. *Design Science, 3*(10). doi:10.1017/dsj.2017.11.

Hay, L., Cash, P., & McKilligan, S. (2020). The future of design cognition analysis. *Design Science, 6*(20). doi:10.1017/dsj.2020.20.

Hoover, S. P., Rinderle, J. R., & Finger, S. (1991). Models and abstractions in Design. *Design Studies, 12*(4), 237–245. https://doi.org/10.1016/0142-694x(91)90039-y.

Hummel, J. E., & Doumas, L. A. A. (2023). Analogy and Similarity. In R. Sun (Ed.), *The Cambridge Handbook of Computational Cognitive Sciences*, 451–473. chapter, Cambridge: Cambridge University Press.

Johnson-Laird, P. M. (1983). *Mental Models*. Harvard University Press.

Johnson-Laird, P. N. (2010). Mental models and human reasoning. *Proceedings of the National Academy of Sciences, 107*(43), 18243–18250. https://doi.org/10.1073/-pnas.1012933107.

Koedinger, K. R., & Anderson, J. R. (1990). Abstract planning and perceptual chunks: Elements of expertise in geometry. *Cognitive Science, 14*(4), 511–550. https://doi.org/10.1207/s15516709cog1404_2.

Koskela, L., Paavola, S., & Kroll, E. (2018). "The role of abduction in production of new ideas in Design", In: Vermaas, P.E. & Vial, S. (Eds.), *Advancements in the Philosophy of Design*, Springer, Cham, Chapter 8. https://doi.org/10.1007/978-3-319-73302-9_8.

Kroll, E., Le Masson, P., Weil, B. (2023). Abduction and Design Theory: Disentangling the Two Notions to Unbound Generativity in Science. In: Magnani, L. (eds) *Handbook of Abductive Cognition*. Springer, Cham. https://doi.org/10.1007/978-3-031-10135-9_47.

March, L. (1976). "The logic of design and the question of value", In: L. March (Ed.), *The architecture of form*, Cambridge: Cambridge University Press, pp. 1–40.

Mitchell, M. (2023, September 10). *Can large language models reason?*. https://aiguide.substack.com/p/can-large-language-models-reason.

Mitchelmore, M.C., White, P. (2012). Abstraction in Mathematics Learning. In: *Seel, N.M. (eds) Encyclopedia of the Sciences of Learning.* Springer, Boston, MA. https://doi-org.ezproxy.neu.edu/10.1007/978-1-4419-1428-6_516.

Nielsen, J. & Landauer, T. K. (1993). A mathematical model of the finding of usability problems. *Proceedings of ACM INTERCHI'93 Conference*, Amsterdam, The Netherlands, 24-29 April 1993, 206-213.

Peirce, C. S. (1933). *Collected Papers of Charles Sanders Peirce,* 2nd ed., edited by C. Hartshorne, P. Weiss, and A. Burks, 1931–1958, Cambridge MA: Harvard University Press.

Saldaña, J. (2009). *The coding manual for qualitative researchers*. Sage Publications Ltd.

Sampson, W. W. L., Khan, S. A., Nisenbaum, E. J., & Kralik, J. D. (2018). Abstraction promotes creative problem-solving in Rhesus Monkeys. *Cognition, 176*, 53–64. https://doi.org/10.1016/j.cognition.2018.02.021.

Shin, S., & Gerstenberg, T. (2023). Learning What Matters: Causal Abstraction in Human Inference. *Proceedings of the 45th Annual Conference of the Cognitive Science Society*. https://doi.org/10.31234/osf.io/br2vz.

Simon, H.A. (1977). The Theory of Problem Solving. In: *Models of Discovery. Boston Studies in the Philosophy of Science, vol 54.* Springer, Dordrecht. https://doi.org/-10.1007/978-94-010-9521-1_13.

Starkey, P., Spelke, E. S., & Gelman, R. (1990). Numerical abstraction by human infants. *Cognition, 36*(2), 97–127. https://doi.org/10.1016/0010-0277(90)90001-z.

Tomasello, M. (2001). First steps toward a usage-based theory of language acquisition. *Cognitive Linguistics*, 11(1-2), 61-82. https://doi.org/10.1515/cogl.2001.012.