Design theory

Product Design and Development (ME30294).
Jérémy Bonvoisin, Dept. Mech. Eng., University of Bath

Last update: October 21, 2019

Abstract

This lecture intends to take a step backwards and question what actually the essence of design is. It introduces some descriptive models of the design activity that can be used to reflect on more practical aspects of design practice. This lecture is hoped to support the student's ability to maintain a reflective dialogue with their design practice and to improve it along their upcoming experience.

Keywords. wicked problems; analogical reasoning; design heuristics; co-evolution; C-K theory

Contents

1 Disclaimer

This lecture is not about mathematical or engineering methods. We will not teach you new practical tricks so you can make cool things. Because this is not what design is. Design is a cognitive activity that builds upon creativity and experience and cannot be broken down into equations and algorithms.

Like Management, design is highly contextual and does not allow 'one-size-fits-all' approaches. Consequently, this lecture may sound more theoretical, even 'philosophical', or 'fuzzy', than what you have been used to so far.

The idea behind this lecture is that–if we can't give you the magical wand of design–we can give you some theoretical background to help you building your own design practice. It aligns with the Donald Schoen's philosophy of reflective practice that considers the ability to reflect on action as a critical factor of professional competence [Slide 1]. This lecture will hopefully give you a spark to fire your own reflective practice and help you building your own design competence.

2 Design: a confusing concept

You're in a course called Product *Design* and Development, led by lecturers on Engineering *Design*. Some of you were in the UG courses "Design 1 to 4". You may have heard about Industrial Product *Design* and Interior *Design*, visited a *design* shop or heard someone saying that something "is pretty good *design*" [Slide 2]. While you may have come across the word 'design' a number of times, there are good chances that nobody ever tried to explain the common denominator of all these occurrences.

What is the common denominator of the occurrences of the word 'design' in these contexts? What does this say about the remit of the *designer*? And what are we referring to in the title "Product Design and Development"? The objective of this lecture is to answer these questions and to give you some hints about what design is, to characterize it, to discuss about the remit of the designer and what makes a good designer.

The task is not made easy by the fact that the word 'design' is generally used to depict very different meanings, as in the illustrative sentence "Design is when designers design a design to produce a design" [Slide 3][1]. In this sentence, the word 'design' is used four times to depict 1) a domain of human activity, 2) an action, 3) an intellectual projection and 4) a physical product—very confusing.

While there isn't *one* definition of design, there are many theories shedding different lights on what design is. One reason for this is that "the activity called design is not fully understood" [2]. It happens for a great part in the designer's head and offers little interface for observation and experimentation. However, there are some characteristics that are very specific to the design activity and that is useful to be aware of as a practitioners.

The objective of this lecture is to mention a relevant subset of these theories in order to produce for you a useful picture of design. These theories provide explanatory frameworks to represent what happens when somebody designs something. They provide a basis to explain some of the oddities of design and justify why some aspects are discussed in other parts of the course. Through this lecture, we hope to help you making connections between the concepts addressed in this course and therewith to ground your life-long design learning process [Slide 4].

3 Let's start with a story

The story[1] says that in 1936, as the tire manufacturer Michelin took over the bankrupt car company Citroën, the newly established vice-president and chief of the Engineering and Design department Pierre Boulanger sent the following design brief to the engineering department: "Study a car that can carry two farmers on clogs, 50kg potatoes or a keg at 60km/h with a fuel consumption of three litres of petrol over 100km. This vehicle should be able to cope with tracks of the worst condition. It should be light enough to be easily driven by a beginner driver and egg baskets carried on the back bench must arrive undamaged at destination. It must be possible to board 4 people without requiring them to remove their hats. Its price should be way below those of the Traction Avant [a previous bestseller car from Citroën]."[Slide 5][2] By 1937, the engineering department came out with a car fulfilling the brief: the TPV ("Très Petite Voiture", very small car in English) [Slide 6]. After some years of delay caused by WWII, Citroën launched 1949 the production of the now renamed and different looking car the 2CV ("deux chevaux", 2 two steam horses in English) [Slide 7]. If the TPV was already fulfilling the brief, what made the car change between 1937 and 1949?

This story is a good illustration of the four ideas we will develop in this lecture [Slide 8]:

- Design addresses so-called "wicked" problems, a concept we will present in the next section. The "wicked" nature of the design problems have three consequences:
  - Design is not something you can automate but rather requires a specific type of thinking we'll call case-based reasoning.
  - Design involves a co-evolution processes where both the problem and the solution are iteratively refined.
  - Design can be best understood as a learning process.

4 Design addresses "wicked problems"

Problems addressed in mathematics or computer science are convenient, because they can be elicited precisely and there is always a way to generate systematically all solutions (provided you have enough time). Take for example the following problem: "Given a 3×3 board with 8 tiles [...] and one empty space. The objective is to place the numbers on tiles to match final configuration using the empty space. We can slide four adjacent (left, right, above and below) tiles into the empty space." [3][Slide 9]. You would be able to define an algorithm that will compute all paths available from the initial situation until you find the final configuration [Slide 10](what is called brute force search).

[1] a good entry point to read the full story can be found on Wikipedia.
[2] own traduction of the original text in French reported here.

Unfortunately, in design like in real life, problems are beyond the reach of exact and exhaustive formulation. 'Only in trivial cases is the computation of the optimum alternative an easy matter. [In] the real world we usually do not have a choice between satisfactory and optimal solutions, for we have only rarely a method of finding the optimum " [4, p. 118-120]. In design like in real life, problems are 'wicked' [Slide 11-12][5][extra reading].

The prime characteristic of *wicked problems** is that they are ill-defined: they cannot be definitively described because of the partly conflicting, ambiguous, unconscious views of the involved stakeholders. "An ill-defined problem is one in which the requirements, as given, do not contain sufficient information to enable the designer to arrive at means of meeting those requirements simply by transforming, reducing, optimizing or superimposing the given information alone. Some of the necessary further information may be discoverable simply by searching for it, some may be generateable by experiment, some may turn out to be statistically variable, some may be vague or unreliable, some may arise from capricious fortune or transitory preference and some may be actually unknowable." [6][extra reading].

This is aspect is reflected in the case of the 2CV [Slide 13]: while engineers came out with a solution that fulfilled the original brief, it became clear that some design criteria have not been considered. There were unconscious (or at best tacit) at the time of writing the brief. One of these criteria may have been the aesthetics. This criterion could have been mentioned in the original design brief, but it would not have been possible to put an objective decision criteria on it, since different people may disagree about what makes a car pretty. Some of you who participated in Design 4 (machine design) may remember that, the more you explored the problem, the more you had questions about design criteria that weren't in the initial brief. A design brief is *always* incomplete and ambiguous, since there will always be some criteria that are either unconscious or not definable objectively. (Corollary: if you can precisely define a problem, then it is no design problem but rather an optimization problem).

In the absence of a finite and objective description of the problem, solutions to wicked problems are neither true, false or optimal. They can be at best fairly good, sufficient or satisfying. You cannot prove that the 2CV was the *best* car Citroën's engineers could design, since there is no objective way to compare cars. You cannot reject the hypothesis that there can be a car that would fit the specifications of the 2CV better than the 2CV.

Because of that, wicked problems do not deliver a way to state objectively when the job is done and the design process can stop. "Even worse, there is no 'solution' in the sense of *definitive* and objective answer" [5][3]. Citroën's engineers could have worked longer and refined the design indefinitely. They just stopped at some point because they found a satisfactory solution that balances time investment and fitness with the design criteria.

[3]Emphasis is not in the original text.

5 Three visions on design

The wicked nature of design problems makes of design "problem solving without a goal" [4, p. 106]. This has particular consequences we will review in the next three sections. Each of them introduces a specific perspective on design that derives from the characteristics of wicked problems.

5.1 Design is case-based reasoning

We mentioned earlier that in mathematics and computer science, problems can be elicited precisely and there is always a way to generate systematically all solutions. We can also find problems that can be defined precisely in theory but cannot be solved in practice. For example, while it is possible to define an exact algorithm to predict the best move to do next in chess, the volume of operations required to terminate this algorithm would certainly hit the limits of available computational power. The traveling salesman problem, which is about finding the shortest path linking a given set of cities [Slide 14], is another example: testing all possible permutations of cities to find the shortest path between all of them becomes impractical even with only 20 cities, since it would require trillions of operations (>1e18).

To solve these non-trivial problems, computer scientists and mathematicians need to find 'tricks', or to use proper vocabulary *"heuristics*"*, in order to come with solutions and arbitrarily cut off parts of the unmanageable space of possible directions to go. A heuristic for the traveling salesman problem is the nearest neighbour algorithm. The idea is to choose as your next destination the city which is the closest to the city you are currently in. This heuristic comes from experience and is known from experience to yield a satisfying solution in a short time, although this solution is often not optimal [Slide 15]. In more general terms, heuristics are "tricks" gained from experience—strategies which have shown to be advantageous in previous problem solving. They only guarantee to generate fairly good solutions, in contrast to exact algorithms which guarantee to find *all* solutions and *the* optimal one.

Like for these complex mathematical problems, it is not possible to enumerate exhaustively all solutions of a wicked problem. Consequently, finding solutions for a wicked problem necessarily requires the use of some kind of heuristics as well.

The first available trick for designers is to go back solutions they know from experience. Designing a new car, they would probably come out with something that looks very similar to cars they know [Slide 16]. Asked to design a new coupé with an 'aggressive' look, they may design a car that looks like an animal they value as aggressive [Slide 17]. Doing so, they use their capacity to build analogies. Analogy is a mechanism we instinctively use in our efforts to understand the world around us or to explain how we understand it. For example, explaining the working principles of electricity often involves a comparison with water flow through a system of pipes [Slide 18]. "*Analogical reasoning**" is based on the idea that problems or experiences outside the one we are currently

dealing with may provide some insight or assistance. [...] Analogy is a way of recognizing something that has not been encountered before by associating it with something that has. [It] can be used in common situations, where the previous experience is directly applicable, or in unique or creative situations, where the previous situation shares something with the new situation, but the differences are just as interesting as the similarities" [7, p. 1]. The way analogical reasoning works in design is that the designer "is reminded of the previously solved problem because it has some relevance to the new problem. After the person recalls a previously solved problem, certain aspects of the previous problem's solution are used in the new context and others are not [Slide 19]. For example, [While designing a 18-meter long pedestrian bridge over a busy street, the previous example of] a pedestrian bridge with a span of 15 meters may be recalled [...]. The same design for the superstructure, such as the steel arch, may be used, but the span and sizes of the steel members will change." [7, p. 1-4]. In short, analogical reasoning is a critical ability to find solutions that involves both experience (the base cases) and creativity (the ability to recall make links between cases). It allows designers to build relations between cases and come out with general rules of action (the heuristics) that help them to generate solutions (for example: "avoid complicated shapes that are complicated to manufacture").

Understanding design as analogical reasoning implies that the more cases a designer has memorized, the more they are likely to find design solutions. Generating "a design for a bridge requires not only an understanding of the analysis of bridges, but exposure to examples of several bridge designs." [7, p. 1]. Ideally, cases are memorized through practice, but can alternatively be gained through learning. Good designers commonly seek for external stimuli to populate their base when they are not actively solving a problem [8], for example by browsing design magazines [Slide 20]. Undergraduate students in mechanical engineering learn machine elements in order to build a base of standard elements they can use, combine, adapt to specific problems through analogical reasoning [Slide 21]. Specific design knowledge can also be passed on in the form of design heuristics. Some researchers in design provided sets of design heuristics collected from designers' practice. Some of them are applicable to any design problem, like the Iowa State University's 77 heuristics or the 40 principles of invention involved in the creativiy method TRIZ. Some other are targeted at specific design criteria, like product flexibility [Slide 22]. These heuristics work as some kind of 'extension' or 'plug-in' of the designer's heuristics base. It extends the corpus of heuristics a designer can use to build analogies with a given design problem and come out with creative solutions. The use of these heuristics in controlled as well as real design settings has been shown to increase the variety of idea generated and hence to support innovation [9]. Exposure to a variety of heuristics also proved to help developing expertise in design [10].

Take-aways of this section:

- *Design problems cannot be computed to find an optimal solution.*

- *Instead, solution search is lead by analogical reasoning, which is fuelled by experience.*
- *Designer's experience grows with exposure to either personally experienced or observed examples.*
- *Designers instinctively infer* design heuristics* *out of examples .*
- *To increase experience: be curious, read design magazines, analyze existing designs, tinker around.*

5.2 Design as co-evolution

Most product development process models (such as the VDI 2221 [11]) are based on the Waterfall model which involves a linear sequence of stages and gates. These models understand design as a problem solving process, where you start with a problem and end with a solution [Slide 23]. They describe design as a black box that takes product requirements as inputs and yields a product definition as output. Ulrich and Eppinger's reference lecture on product design and development states it clearly: "In an ideal world, the team would establish the product specifications once early in the development process and then proceed to design and engineer the product to exactly meet those specifications." [12, p. 73]. In other words, you *first* define the problem, and *second* design a solution for this problem.

But this sentence begins with "in an ideal world" because it acknowledges that the reality is more complex and that problems do not precede solutions in design, although it would be way easier to handle product development as if it would be the case.

More than the simple search of a satisfactory solution to a given problem, design is about *refining a problem* and a solution at the same time. In design, "the 'problem' is [...] not the statement of requirements. [... It] is *obscurity* about the requirements, the practicability of envisageable provisions and/or misfit between the requirements and the provisions" [6][4] [key reading]. Design is about reducing uncertainty about "what designers need to know about the problem" and which "only becomes apparent as [they] are trying to solve it" [13].

All this means that the design activity involves a *co-evolution** of the problem and the solution [Slide 24][14]. "[The] formulation of the problem at any stage is not final [...]. As the design progresses, the designer learns more about possible problem and solution structures as new aspects of the situation become apparent and the inconsistencies inherent in the formulation of the problem are revealed. As a result, [...] the problem and the solution are redefined" (*ibid.*, p. 5). Consequently, it appears that design involves an "iterative interplay to 'fix' a problem from the problem space and to 'search' plausible solutions from the corresponding solution space" (*ibid.*, p. 1). Externalising the design (e.g.

[4] Emphasis is not in the original text.

sketching it) allows a dialogue between the problem and solution to evolve, it allows questions to be raised about the width of the solution space and the range of acceptable requirements, what in turns allows decisions to be made to reduce those spaces and to converge towards a fixed design.

In summary, design is about refining *both* a problem *and* a compatible solution to this problem [Slide 25]. Consequently, the design activity does not start with precise product specifications but produces them out of rough "hopes and aspirations", eventually gained from a study of customer needs [12, p. 73]. The exit condition of this interactive process does not only lie in the consistency between the problem and the solution (when the product design is said to "meet" the specifications or "fulfill" the requirements), but in external factors: for example, time and money went out, or the team is satisfied with the degree of precision achieved in the formulation of both problems and solutions.

Take-aways of this section:

- *The word of saying "clearly stating the problem is half the way to solve it" also apply to design.*
- *In most cases it is not realistic to expect specifications can be all set at the beginning of the design process.*
- *Do not accept the brief 'as given', your role as designer is to question it.*

5.3 Design is a learning process

Since there is no fully defined 'problem' in design, we cannot really define design as a problem solving activity. Instead, we referred to design as an 'exploration' process. This exploration is aimed at reducing 'obscurity' regarding the considered object (the requirements, the system which is supposed to fit with these requirements, the context in which this systems is supposed to take place). In other words, it is a *learning process* through which a designer attempts to get a complete and logically consistent picture, a picture that makes sense.

This view of the design activity fits well with the *C-K theory** introduced by Hatchuel and Weil [15][5]. This theory defines design as the parallel expansion of two ensembles: the knowledge and concept space [Slide 26].

The "knowledge space" K is defined as the "space of propositions that have a logical status" for a given designer or team of designers. In other words, K is the set of all propositions a designer or a group of designers know to be true or false. The elements of this space are propositions designers can rely on and don't have to explore because they know these to be true or false. For example, a designer may know that knives are made to cut and airplanes can fly, they can take these propositions for granted.

The "concept space" C is defined as the space of propositions that have no logical status. In other words, C is the set of propositions a designer or the

[5] All quoted sentences in this section are taken from this source ([15])

group of designers do not know whether they are true or false. For example, a designer may never have experienced a knife that can fly or a plane that can cut. The propositions "a knife that can fly" and "a plane that can cut" are neither true or false for them as long as they don't experience these or get proven it is not possible to do such things. Propositions of C are hypothetical ideas thrown in a design process waiting to be validated and transformed into knowledge. These are the creative bits of design.

From this, design is defined as a sequential process of cross-fertilization between C and K spaces through four types of operations building the so-called "design square" [Slide 27]:

- $K \rightarrow C$: this operator creates elements in C (new concepts) from elements of K (bits of knowledge). A concept can be generated by combining elements of knowledge. A designer knowing the proposition "airplanes can fly" and "there is a planet called Venus" could come to the proposition of an "airplaine flying on Venus". This proposition remains a concept (that is, an element of C) as long as he has not been proven whether it is possible to make an airplane fly on Venus or not (the answer here).

- $C \rightarrow K$: this operator creates elements in K (bits of knowledge) from elements of C (concepts). Transforming concepts into knowledge corresponds to what is generally called 'concept validation'. This can be done in various ways, such as "consulting an expert, doing a test, an experimental plan, a prototype, a mock-up". An element of C is transformed in an element of K when it is given a logical status (true or false).

- $C \rightarrow C$: this operator creates concepts out of concepts. It amounts to jump onto an idea to create a new one.

- $K \rightarrow K$: this operator creates knowledge out of knowledge. This brings us back to the conventional rules of deductive reasoning used for proving mathematical theorems or in syllogistic reasoning such as: "All men are mortal / Socrates is a man / Therefore, Socrates is mortal."

Elements of K are by convention considered to be unordered while elements of C are represented in a tree. The root of the tree is the initial concept, that is, the concept the design process starts with (e.g. we want to design a "knife that can fly"). All other elements of the concept tree are more detailed and alternative concepts derived from that initial concept along the exploration process. Each new level in the tree is an added level of detail in the expression of the initial concept. Different elements at the same level in the tree are different alternatives for the same level of detail. Generating these concepts implies either extending a previous concept ($C \rightarrow C$) or recalling an element of knowledge from K ($K \rightarrow C$) to be combined with an existing concept. New elements may also appear in K as the result of deductive reasoning ($K \rightarrow K$) or of the validation of a concept ($C \rightarrow K$). The design process stops when a path from the root to a leaf of the concept tree delivers a picture which fully "makes sense". It delivers

the picture of a fully detailed design—something which can be built, something that *exists*, has a logical status, and can be fed back into K ($C \rightarrow K$).

The case of a "nail holder avoiding to hurt one's hand while hammering" reproduced here is reported in [16]. The reasoning process starts with the concept {safe knocking a nail} [Slide 28], which is a combination of the elements of knowledge {safe} and {knocking a nail}. The designer or the design team know about {Knocking a nail} and can rephrase it as {hammer in right hand, nail in left hand, energy given by shocks}. Having recalled the element of K {hammer}, the designers can choose to challenge it and explore two refined concept alternatives {Safe knocking a nail with a hammer} and {Safe knocking a nail without a hammer}. As well, having recalled the element of K {nail in left hand}, they can challenge it and explor two further refined concept alternatives {Safe knocking a nail with a hammer and with the nail in the left hand} and {Safe knocking a nail with a hammer and without holding the nail in the left hand}. The designer team also knows about {safe} meaning {without knocking on fingers}. From this, they can derive alternative concepts to hold the nail without having the risks of knocking it with the hammer, such as {avoid the hammer to derive from the desired trajectory} or {protect the hand from the hammer}. These concepts can be refined into more precise concepts {a trajectory control device} and a {hilt} or {protecting glove}. The same tree-shaped recursive search algorithm can be continued from any element from C or K, such as {safe hammering with hammer in right hand and left hand not holding the nail} [Slide 29].

Take-aways of this section:

- *Design is a process of expansion of the designer's knowledge and available concepts.*
- *Design terminates when concepts of interest are turned into kwowledge, that is, their feasibility has been proven and they can be turned into reality.*
- *Concepts are created by recombination of elements of knowledge.*

6 Outlook - On designer's skills

Nigel Cross, a British design researcher, observed actual practices of designers and introduced design thinking as the cognitive process and abilities at play in the design activity. He identified three fundamental competences which seem to be common across exceptional design practitioners [Slide 30-32][13, reformulated by Elies Dekonkinck] and fit well with the theories provided in this lecture:

- "Framing the problem in a distinctive and sometimes rather *personal way*: do not accept the brief 'as given'. The aim is to *question it* and go beyond it or steer towards a *greater opportunity*". 'Questioning the brief' is part of the idea of design as a learning process: requirements offer an incomplete

picture of the problem and require exploration to reduce their 'obscurity'. This exploration enables making unexpected discoveries, eventually leading to reframe the design process towards 'greater opportunities'. Accepting the brief as given bears the risk of missing these opportunities and yielding suboptimal, that is, poor, design. Designers necessarily understand the design problem in a 'personal way', according to their own knowledge space K which has been populated along their personal or professional experience. The richer this knowledge space is, the greater the ability to reframe the problem in an original way.

- "Taking a broad systems approach to the problem. Going beyond the part or object to be designed, designing the system around it as well, or at least understanding the whole system context in depth". This statement implies that the picture delivered by the design brief is not only incomplete: it also a point of view, where some objects are masked behind other objects. The deeper the exploration process goes, the deeper the understanding a designer gets, the broader the considered system is, the larger the chances are that the fundamental aspects of the problem are covered, and the more optimal the design can become. We'll come back to this in the lecture on systems thinking.

- "Designing from first principles: sometimes means designing from the fundamental physics or using minimalist approach such as 'form follows function"'. Design principles are another word for design heuristics. Designers follow principles of action they believe in or came across along their experience (such as 'form follows function'). In the absence of a mathematical formula for generating good designs, this is their way to populate the decision tree of problem solving or the concept space C efficiently.

Credits

References

[1] J. Heskett, "Past, present, and future in design for industry," *Design issues*, vol. 17, no. 1, pp. 18–26, 2001.

[2] I. Hybs and J. S. Gero, "An evolutionary process model of design," *Design Studies*, vol. 13, pp. 273–290, July 1992.

[3] Aditya Goel, Kirti_Mangal, and Akanksha_Rai, "8 puzzle Problem using Branch And Bound," May 2016.

[4] H. A. Simon, *The Sciences of the Artificial*. MIT press, 3 ed., 1996.

[5] H. W. J. Rittel and M. M. Webber, "Dilemmas in a general theory of planning," *Policy Sciences*, vol. 4, no. 2, pp. 155–169, 1973.

[6] B. Archer, "Design as a discipline," *Design Studies*, vol. 1, no. 1, pp. 17–20, 1979.

[7] M. L. Maher, M. B. Balachandran, and D. M. Zhang, *Case-Based Reasoning in Design*. Psychology Press, 2014.

[8] M. Gonçalves, C. Cardoso, and P. Badke-Schaub, "Inspiration choices that matter: The selection of external stimuli during ideation," *Design Science*, vol. 2, Jan. 2016.

[9] S. Yilmaz, S. R. Daly, J. L. Christian, C. M. Seifert, and R. Gonzalez, "Can experienced designers learn form new tools? A case study of idea generation in a professional engineering team," *International Journal of Design Creativity and Innovation*, 2013.

[10] S. Yilmaz, S. R. Daly, C. M. Seifert, and R. Gonzalez, "How do designers generate new ideas? Design heuristics across two disciplines," *Design Science*, vol. 1, Nov. 2015.

[11] VDI, "Design Handbook 2221, Systematic Approach to the Development and Design of technical Systems and Products," 1993.

[12] K. Ulrich and S. Eppinger, *Product Design and Development*. New York: McGraw-Hill Higher Education, 5 ed., Aug. 2011.

[13] N. Cross, *Design Thinking: Understanding How Designers Think and Work*. Berg, 2011.

[14] M. L. Maher, J. Poon, and S. Boulanger, "Formalising Design Exploration as Co-Evolution," in *Advances in Formal Design Methods for CAD*, IFIP — The International Federation for Information Processing, pp. 3–30, Springer, Boston, MA, 1996.

[15] A. Hatchuel and B. Weil, "A new approach of innovative Design: An introduction to CK theory.," in *DS 31: Proceedings of ICED 03, the 14th International Conference on Engineering Design, Stockholm*, 2003.

[16] A. Hatchuel, P. Le Masson, B. Weil, *et al.*, "CK theory in practice: Lessons from industrial applications," in *DS 32: Proceedings of DESIGN 2004, the 8th International Design Conference, Dubrovnik, Croatia*, pp. 245–258, 2004.