

DESIGN THINKING UNPACKED: AN EVOLUTIONARY ALGORITHM

Janne KORHONEN¹, Lotta HASSI²

¹S.E.O.S. Design Ltd

²Helsinki University of Technology

ABSTRACT

“Design thinking” has been proposed as a method for achieving better solutions in fields as diverse as architecture and organizational design. Despite various definitions for design thinking, there have been relatively few articles that aim to explain on a general level how and why, and under what preconditions, design thinking actually works. This paper presents a hypothesis that design thinking can be understood as a generic problem-solving algorithm whose qualities can be compared to other generic algorithms. The similarities between design thinking and evolutionary optimization algorithms suggest that design thinking is among the best possible algorithms for seeking solutions to ill-defined problems where there is no single obvious “best” solution. This helps to better define what design thinking is and leads to practical implications for product development managers and designers.

Keywords: design thinking, evolutionary optimization, evolutionary algorithm, problem solving, NPD

1 INTRODUCTION

Recent literature in design practices has highlighted the benefits of multidisciplinary product development teams, human-centred exploration of user’s real needs, and early and iterative prototyping of ideas and concepts. Often called “design thinking,” proponents of the method claim that resulting inductive-deductive-abductive method works much better than previous, primarily deductive methods of R&D problem-solving in fields ranging from education to engineering (e.g. Brown, 2008). However, the benefits of “design thinking” approach are often unsupported by theoretical background. This study makes an attempt to build a theory of real-world new product development (NPD) processes¹ as problem-solving algorithms, while comparing them to problem-solving algorithms whose effectiveness can be theoretically examined. In effect, this study examines whether or not, and under what conditions, “design thinking” is the most effective problem-solving tool available to NPD teams.

The approach used is to look at NPD process through the lens of evolutionary optimization theory. Although evolution has been often seen as a metaphor for design and particularly technological development (see Ziman, 2000 for an overview and further references) and some authors have viewed design practices as evolutionary (e.g. Langrish, 2004; Whyte, 2007), it can be argued that instead of directly mapping biological into technical, the discussion should move to a higher level of generic *algorithms*. As an example of more modern approach, Beinhocker (2006) has elegantly explained technology S-curves (Foster, 1986) and disruptive innovations (Christensen, 1997) as results from evolutionary algorithm in action. This study builds on Beinhocker’s work and attempts to explain NPD process as an example of evolutionary algorithm, and develop practical implications for designers and managers.

The paper is organized as follows: first, product development is defined as a search for optimal design from a hypothetical “design space.” Second, theoretical properties of design spaces are discussed. Third, theoretical search strategies are briefly introduced and evaluated. Similarities between search strategies and design approaches are discussed. Fourth and finally, practical implications are briefly outlined.

2 NEW PRODUCT DEVELOPMENT: A SEARCH THROUGH A DESIGN SPACE

Generalized models of product development process have been developed by numerous authors (see

¹(Although this text refers to new *product* development for reasons of clarity, the principles apply to other development projects as well.)

e.g., Brown and Eisenhardt, 1995; Ulrich and Eppinger, 2008), and what is clear is that NPD process seeks to develop knowledge about technologies, markets, and users, and apply that knowledge towards more effective and efficient products and services. Thus, product development is fundamentally about searching for the “blueprints” of the best possible design. These blueprints can be thought to be housed in a practically infinite library of blueprints for everything - to paraphrase Dennett (1995), the “Library of All Possible Designs.” Within this library, blueprints for all the different combinations of all the parts, materials, manufacturing processes and usage scenarios can be thought to exist, from stone axes to spaceships. This library is referred in evolutionary theory as a *design space* for all possible designs (Dennett, 1995, pp. 107-113). Design spaces can be constructed for anything that can, in principle at least, be memorialized in some form and transmitted to others (Dennett, 1995; Beinhocker, 2006). Design spaces can be constructed in smaller or larger variations: for example, a design space for combining two Lego blocks with each other is relatively small, while a design space for all desktop computers is enormous. As can be imagined, the huge majority of the entries in this library are outright impossible to realize and/or completely useless; others are just tiny variations of existing - or future - designs. To determine whether one design is better than some other, the design space is visualized as a *fitness landscape* (Wright, 1932; Kauffman, 1993). By first arranging the design space so that the distance between two designs represents the differences between the designs, and then assigning a value of fitness for each design, we can imagine a landscape of peaks and valleys. Good designs can thus be thought of as high peaks in the fitness landscape, and the problem of finding good designs in the near infinity of design space then becomes a problem of finding high peaks in the fitness landscape. (Figure 1)

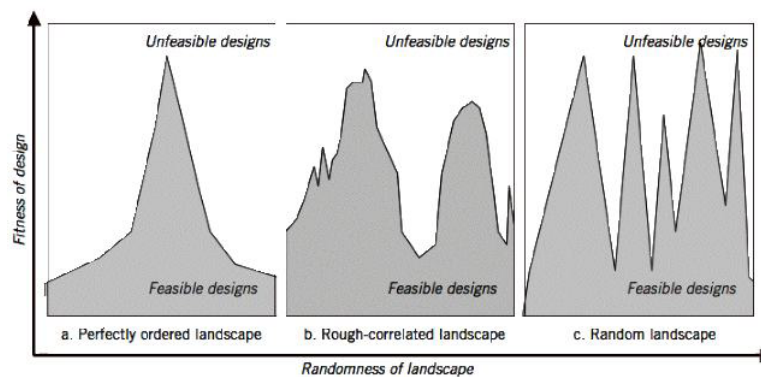


Figure 1 Examples of fitness landscapes

Fitness is defined here as fitness for purpose. A design is considered to have better fitness than another design if it performs its intended function more efficiently or effectively. Fitness is a dynamic function, and what is fit at one point of time may not be fit some time later. Typically, designs have a *minimum fitness* that determines, for example, whether the design does what it should do. (Beinhocker, 2006). Potentially, two extremes for the fitness landscape are possible: first, that there is no correlation between the difference of two designs from each other and their fitness, resulting to a completely random landscape (Fig. 1c). The second extreme is a perfectly ordered landscape where a single design is most fit and the fitness of designs becomes lower as their distance from that optimum design increases (Fig. 1a). In the real world, most fitness landscapes are somewhere in between (Kauffman, 1995, pp. 161-189) (Fig. 1b). Two designs that differ in very small details - say, a PC with either grey or beige casing - are likely to be similar in fitness. However, some small changes may have very large effects (for example, changing the measurements from metric to Imperial is a very small change in terms of information changed), while some large changes may only have a small effect (e.g. Intel and AMD processors are different internally but work equally well). A design space in which small changes in design lead to small or no changes in fitness, but where some small changes have large effects is called *rough-correlated* (Beinhocker, 2006). Rough-correlated landscapes are the standard in biology and in most technologies, and the result is important because what is the most efficient algorithm for the search depends on the shape of the fitness landscape.

3 SEARCHING THROUGH THE FITNESS LANDSCAPE: SEARCH ALGORITHMS

How, then, can this library of all possible designs be searched in a most efficient manner? In keeping with the two extremes of fitness landscapes, two extreme search strategies can also be identified. The search strategy can be completely random, with previous iterations having no effect on future iterations. In product development terms, this would amount to generating random patterns and building prototypes from those patterns, and then discarding the pattern and trying again if the prototype didn't work out. This is obviously a high-risk strategy, which isn't practiced deliberately in the real world.

The search can also be based on random mutation hill-climbing (Mitchell, 1996, p. 116) or *adaptive walk* (Beinhocker, 2006). In adaptive walk, the searcher would take a step in a random direction in the fitness landscape. If the step leads upwards, the searcher would take another step; if it didn't, the searcher would return and take another random step. This strategy is effective for scaling the individual peaks in the fitness landscape. In product development terms, adaptive walk is analogous to perfecting - often deductively - the existing technology via incremental improvements, but without looking for alternatives in technologies or user needs. Examples of (mostly) adaptive walk strategies in real life are all R&D projects based primarily on applying theory to find out what is possible and how, and only then realizing the design - for example, increasing the packing ratio in integrated circuits. However, if the "landscape" to be searched is rough-correlated, both extreme strategies are likely to lead to suboptimal results. Since there are many more unworkable designs than good designs, the odds are that random search will not find fitness peaks. On the other hand, adaptive walk may find a feasible design, but, unless there is an option of retracing the steps and starting over again, it probably gets stuck in a sub-optimal fitness peak. Of course, real-world NPD processes utilize several strategies simultaneously: humans should be able to use their reasoning facilities to determine when they are stuck in a "fitness peak" and then move on to some other avenue of approach that promises better results, i.e. higher peaks. In terms of fitness landscapes and search strategies, this would amount to going back down, once a fitness peak for a particular approach has been reached.

But going back may be difficult since many processes are path-dependent, meaning that choices made during the process limit what options are - or are perceived to be - available later. Choices made early during the process can, for example, determine the technologies used, and once a choice is made, it may be difficult to question. Whether path dependency is real or just perceived, the end result is similar: once a fitness peak is reached, NPD teams have to consciously wrest themselves to seek for other, possibly higher, fitness peaks, just as firms have hard time switching technologies even though newer technology promises better future returns (as elaborated by e.g. Christensen, 1997). This phenomenon is familiar to many NPD professionals: changing major design features halfway through the process is difficult due to the attachment team members have to something they've been working hard to create, even if the proposed design is clearly underperforming. Market and business demands count, too: even if the proposed design isn't really what is needed, descending the fitness peak and trying to find another way up may be impossible due to financial pressures and plain organizational inertia. Retracing the path is often impossible because doing so would mean relinquishing investments in a design that is still somewhat successful, even though it may be clear that a competing technology would be better in the long run.

4 DESIGN THINKING: THE BEST SEARCH STRATEGY FOR ROUGH-CORRELATED LANDSCAPES

However, once NPD is defined as a search for knowledge about technologies, users and markets through rough-correlated fitness landscape of all possible designs, mathematical tools can be utilized to determine the quality of search strategies. The best strategy turns out to be, in general terms, an evolutionary algorithm (e.g. Kauffman, 1993; Mitchell, 1996). Evolutionary algorithms are a class of search algorithms that combine adaptive walk and random search: evolutionary search goes uphill towards the fitness peak, but also randomly tries other locations to see if they happen to harbor more promising peaks. (Figure 2.)

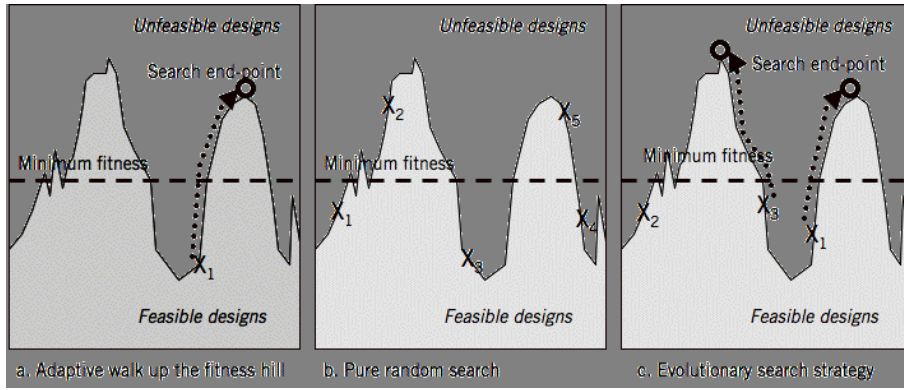


Figure 2 Different search strategies. $X_{1...n}$ represent exploratory search, dotted lines exploitative search

Most innovation-related activities resemble evolutionary search algorithms to a greater or lesser degree. By combining iterative tinkering and testing (inductive) with analysis (deductive) and sprinkling some imagination (abductive thinking), they proceed through both radical (randomly testing other locations for more promising ways uphill) and incremental (going uphill if there's a way) improvement. Iterative tinkering remains essential to the development of innovations, even as the advances in science and technology have improved our capabilities for deduction: most complex design problems remain deductively intractable and are likely to do so, but whether to use primarily inductive or primarily deductive methods depends on the nature of the problem. Well-defined problems that have no ambiguities can be solved through deductive analysis; ambiguous or ill-defined problems with no clear boundaries need inductive (and abductive) thought and tinkering. (Figure 3.)

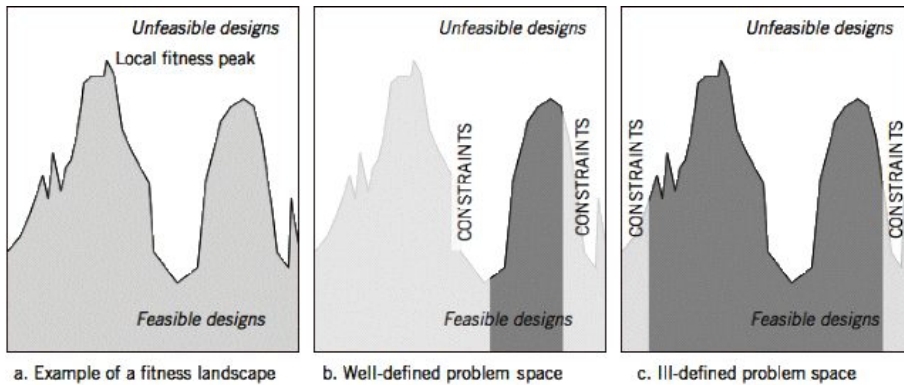


Figure 3 Examples of well- and ill-defined problem spaces

But how are design thinking and evolutionary methods connected? To understand the connection, we need to go back to the definition of NPD as a knowledge-generating process. The goal of the process is to generate knowledge that satisfies the needs of process stakeholders, with economic performance of the resulting product being one of the most obvious fitness criteria. Traditionally, under different “waterfall” and other non-integrated models of product development, fitness of knowledge thus produced is ultimately tested when the product reaches the markets; under design thinking and other integrated, human-centred, iterative approaches (Brown, 2008), the validity and fitness of knowledge is tested much earlier in the process, when corrections can still be made. In terms of fitness landscapes and search strategies, traditional (non-integrated) NPD models represent search strategies that have mostly adaptive walk features. Because length of each “step” is rather long (as knowledge’s fitness is validated only very close to market launch), the step is usually taken into a “safe,” known direction. There is probably at least some iterative tinkering, but exploration is limited and “far-out” possibilities are rarely if ever explored.

On the other hand, iterative and creative multi-disciplinary approaches have much more random jumps built into them. Deductive, *exploitative* search (often by engineers) is used to incrementally improve the quality of knowledge and find the peaks of “fitness hills” scouted out by more *exploratory* search (often by designers), resulting in evolutionary search through fitness landscape. Because the process is human-centered and rough prototyping is encouraged, knowledge’s fitness is often tested much earlier than in other models; because the process focuses on user needs, sometimes the team may see other fitness hills (e.g. alternative ways of solving the user’s problem) than just the one where they start.

5 PRACTICAL IMPLICATIONS

One of the most important implications is that the fitness landscape of a problem defines whether evolutionary or adaptive search is more suited to the situation. In practical terms, this suggests a provisional theoretical explanation why design thinking and other methods which are weighed towards exploration rather than exploitation are good at ill-defined problems, but not cost-effective for problems that have heavily ordered fitness landscapes i.e. that are solved primarily through deductive methods. This result, if it can be confirmed by further study, would be an important qualification to the somewhat nebulous concept of “design thinking” and helps to focus further studies on the subject. Second implication, flowing directly from the first, is that if the fitness landscape of a NPD project is possible to predict or estimate to some extent, it should be possible to decide whether the balance of methods used in the NPD process should be on exploratory (inductive) or exploitative (deductive) methods.

Third important implication of the theory is the effect that the fitness landscape has on product development project. Since many fitness landscapes represent steep hills, progress in a typical project is initially slow, accelerates after a while, and then peaks out in an S-curve-like manner. When the curve flattens out and a local fitness maximum is reached, the NPD project tends to lose momentum, even though other fitness peaks might be worth exploring as well. This is, in effect, a theoretical explanation for practices such as “dark horse” used at Stanford University’s E310 product development project: by forcing the team to develop, for a while, a concept rejected early in the process (the “dark horse”), the management ensures that at least one other fitness hill is explored. Practical implication for NPD managers is that the point to guide the team towards alternative approaches is not only at the beginning of the project, but also at the point where the S-curve of accomplishment evens out. Fourth, since different fitness functions define their own design spaces, different parts of the project may benefit from different search strategies. For example, technology for widgets may be developed through mainly exploitative search, while business plan associated with selling the widgets may require more exploratory approach.

REFERENCES

- BEINHOCKER, E.D. 2006 *The Origin of Wealth*. Boston: Harvard Business School Press.
- BROWN, S.L. and EISENHARDT, K.M. 1995. Product development: Past research, present findings, and future directions. *Academy of Management Review* 20(2), pp. 343-378.
- BROWN, T. 2008. Design Thinking. *Harvard Business Review*, June 2008, pp. 84-92.
- CHRISTENSEN, C.M. 1997. *The Innovator’s Dilemma*. Boston: Harvard Business School Press.
- DENNETT, D.C. 1995. *Darwin’s Dangerous Idea*. New York: Touchstone.
- FOSTER, R. 1986. *Innovation: The Attacker’s Advantage*. New York: Summit Books.
- KAUFFMAN, S. 1993. *The Origins of Order*. New York: Oxford University Press.
- KAUFFMAN, S. 1995. *At Home in the Universe*. New York: Oxford University Press.
- LANGRISH, J.Z. 2004. Darwinian Design: The Memetic Evolution of Design Ideas. *Design Issues* 20(4), pp. 4-19.
- MITCHELL, M. 1996. *An Introduction to Genetic Algorithms*. Cambridge, MA: MIT Press.
- ULRICH, K.T. and EPPINGER, S.D. 2008. *Product Design and Development* (4th Ed.). Boston: McGraw-Hill.
- WHYTE, J. 2007. Evolutionary Theories and Design Practice. *Design Issues* 23(2), pp. 46-54.
- WRIGHT, S. 1932. The roles of mutation, inbreeding, crossbreeding and selection in evolution. *Proceedings of The Sixth International Congress of Genetics*, vol. 1, pp. 356-366. Ithaca, NY.
- ZIMAN, J. (Ed.). 2000. *Technological Innovation as an Evolutionary Process*. Cambridge: Cambridge Univ. Press.

Corresponding Author Contact Information

¹Janne KORHONEN
S.E.O.S. Design, c/o Design Factory
P.O. Box 7700, FI-02150 TKK, Finland
janne.korhonen@seos.fi
+358 41 501 8481
www.seos.fi

²Lotta HASSI
Helsinki Univ. of Tech., Design Factory
P.O. Box 7700, FI-02150 TKK, Finland
lotta.hassi@tkk.fi
+358 50 511 2962
www.decode.tkk.fi

EAD09/075