Scandinavian Journal of Information Systems

Volume 30, No. 2

Publication date: 31 December 2018

eISSN 1901-0990

Reflection note

Design. Building the Solution or Framing a Problem?

Reflections on design science

Jacob Nørbjerg
Copenhagen Business School
jno.digi@cbs.dk

1 Introduction

I appreciate Richard Baskerville's clear and concise summary of the origins of Design Science in Herbert Simon's work. Design—developing possible actions or solutions—is, according to Simon, one of three phases in problem solving or decision making. It is preceded by a search for situations or problems needing attention—the *intelligence* phase—and followed by the *choice* of solutions among the ones identified during design. Simon—and Baskerville—emphasize that the three activities are not to be carried out in any specific order. Problem solving is a non-linear activity.

The purpose of design is to discover and elaborate possible courses of action—or solutions—that will help changing a situation from the present state to a desired end-state. However, the designer (or problem solver) has incomplete and uncertain information only, and the process is, therefore, unlikely to result in an optimal solution, only one or more satisfying solutions. Furthermore, the design process itself is an incremental process, where development and assessment of ideas and suggestions result in more, and more accurate, information.

I have no issues with the account of design and problem solving above. I concur that problem solving is not about choosing the optimal among a set of given alternatives, but creating (designing) possible solutions and choose the one that satisfies some criteria well enough. I also agree that the development of possible designs (solutions) is an incremental search, construct, and evaluate process. What I do find problematic, however, is the account of the design process itself as proceeding top-down from the abstract and general to the concrete aspects of the solution.

> [T]he early stages of search take place in highly simplified spaces that abstract most of the detail from the real-world problem, leaving only its most important elements in summarized form. When a plan, a schematized and aggregated design, has been elaborated in the planning space, the detail of the problem can be reintroduced, and the plan used as a guide in the search for a complete design. (Simon 1972, p. 172, quoted in Baskerville 2018)

I also have issues with the assumption that problems and (desired) solutions are static or given, and that the design process, therefore, "aims to remove the particular differences between desired and present states" (Simon 1996, p. 122, quoted in Baskerville 2018). Problems (and hence solutions) are not given or static, they are perceived differently by different designers and other stakeholders, and they are constantly reframed as the design process itself produces new insights (Checkland and Scholes 1990; Schön 1983; Malhotra 1980; Hirschheim and Klein 1989).

I will use examples and debates from our own field—programming and Information Systems Development—to illustrate our points. I begin with an examination of the almost 50-year old debate between Peter Naur and Niklaus Wirth about how to develop computer programs. From here, I move on to the question of design—or problem solving—in the area of Information Systems Development (ISD). I conclude with some reflections about the interwoven nature of design and use in today's world of agile ISD. Does it make sense to talk about design as an activity distinct from use, and of decisions as static on a world of software platforms and agile ISD, where components are changed or replaced at short intervals?

2 Writing computer programs. Stepwise refinement or messy search?

The early 1970s witnessed an interesting debate between two distinguished Computer Science professsors and expert programmers:. Niklaus Wirth—creator of the programming language Pascal—and Peter Naur—the first professor of computer science in Denmark and one of the creators of the programming language Algol 60. The debate began when Wirth published an article in Communications of the ACM about how to develop solutions to computational problems—AKA computer programs—in a structured and systematic way (Wirth 1971).

Wirth used the 8-Queens problem to illustrate his approach. In the 8-Queens problem one must place 8 chess queens on a standard 8 x 8 chessboard so that all queens are safe from being taken by another queen. In the article, Wirth demonstrated how to use *stepwise refinement* from high level problem descriptions to executable program statements in order to write a program that solves the problem:

> In each step, one or several instructions of the given program are decomposed into more detailed instructions. This successive decomposition terminates when all instructions are expressed in terms of an underlying computer or programming language, and must therefore be guided by the facilities available on that computer or language. ... Every refinement step implies some design decisions. It is important that these decisions be made explicit ...” (Wirth 1971, p. 221)

For the first level of refinement Wirth divided the problem into five sub-problems expressed as three high-level program instructions and two conditional statements (names altered for clarity): *attempt to place queen number j, advance to queen j+1, regress to queen j*-1, *all queens placed, impossible to place all queens.* Each of these were then broken down into more specific statements until the program was complete[1].

There are several similarities between this account and Simon’s description of general problem solving: Problem solving (or programming) moves systematically from general considerations of the problem, over abstract design decisions (sub-problems), to details of the solution (the program). Backtracking in case of failure follows the same path in reverse order. The problem itself is not discussed. The process can be illustrated as an abstract tree where the root represents the original problem, the nodes the intermediate design decisions, and the leaves the detailed program instructions, which, when put together, form the final program. The solution process traverses the tree from root to leaves, and backtracks when a path leads to an unsatisfactory result.

Wirth’s (prescriptive) account of (program) problem solving was immediately questioned by Naur (1972). Following Wirth’s suggestion to try and solve the 8-Queens problem before reading his paper, Peter Naur produced a diary of his solution process, which was distributed over a calendar month. His analysis of the diary demonstrated a process very diffent from the one recommended by Wirth. Staying with the tree metaphor, his process resembled a search through all parts of the tree in apparently random order. His process moved from root to leaves and back again, jumped between different parts of the tree, explored ideas in parallel, proceeded in leaps and bounds, etc. He would also repeatedly return to the problem itself, asking, for example, if the program should find one or all configurations of queens; if symmetrical configurations were to be considered as equivalent or different, and how to present the configurations found by the program to the user?

Naur, like Simon, describes the search for a solution to a (fairly) bounded and static problem, but their processes differ. Simon's account of the design process resembles Wirth's insofar both assume that the designer will move systematically from the abstract and general parts of the solution to the concrete, although Simon acknowledges the incremental and evolutionary nature of solution development (Baskerville 2018; Dahlbom and Mathiassen 1993, chapter 5). Naur also develops his solution in incremental steps, but his process is far from a systematic movement from the abstract to the concrete. He moves between understanding—and revising—the problem, to very detailed program alternatives, and then discovers that the choice of a detailed—low level—solution depends on how he understands the problem, or has implications for choices made in other parts of the program.

3 IS development. What is the problem?

The 8 queens problem is a bounded and fairly well defined programming problem, and the solution (the program) has little or no ramifications beyond the questions about problem solving it raises. Naur's repeated rumifications over the problem did not, after all, question the underlying rules of Chess or the relevance of the 8-Queens problem, and his revisions of the problem specification did not have far reaching implications.

Moving from the realm of programming problems to IS development, however, makes us question the problem itself. Information Systems include people and organizations, and have ramifications far beyond the computer program itself (Markus 2004; Hirschheim and Klein 1989). Problems in IS development projects are fuzzy and ill-defined. There are no given problems in social settings but competing interpretations of a situation, each leading to different visions of change (Hirschheim and Klein 1989; Markus 1983). And implementing a solution leads to new discoveries about the problem (Checkland and Scholes 1981).

4 Collaborative design in use

Simon wrote at a time where the timespan from problem identification (requirements specification) to solution (IS implementation) was large. Today's information systems are, however, not created in one big leap but released in small incremements and constantly modified. Particularly in the web arena.

New or modified apps are added to exitising platforms at a dizzying pace as users of existing solutions discover new opportunities or needs. Does it then, make sense to talk about design—or problem solving—as distinct from use as Simon does, or are these activities even more intertwined than I have discussed above?

And are we—as IS professionals—supposed to find the solution to *the* problem or engage in dialogue with stakeholders to help them describe *their* problem(s)?

Finally, is the goal of DSR to develop a theory of *a* design or a theory of engagement in mutual and collective discovery of problems?

Notes

1. Space does not permit a complete account of Wirth's solution process. I refer the reader to Wirths paper or to the excellent description and discussion in (Dahlbom and Mathiassen 1993, chapter 4)

References

Baskerville, R., (2018): The emergence of design science research from decision theory. *Scandinavian Journal of Information Systems*, (30:2): 55-62.

Checkland, P., Scholes, J., (1990). *Soft systems methodology in action*. Chichester, Wiley.

Dahlbom, B., and Mathiassen, L., (1993). *Computers in Context. The Philosophy and Practice of Systems Design*. Malden, MA: Blackwell.

Hirschheim, R., and Klein, H., (1989). Four Paradigms of Information Systems Development. *CACM* (32: 10): 1199-1216.

Malhotra, A., Thomas, J. C., Carroll, J. M., and Miller, L. A., (1980). Cognitive Processes in Design. *International Journal of Man-Machine Studies* (12): 119-140.

Markus, L. M., (1983). Power, Politics, and Mis Implementation. *CACM* (26: 6) : 430-444.

Markus, M. L., (2004). Technochange Management: Using It to Drive Organizational Change. *Journal of Information technology* (19:1): 4-20.

Naur, P., (1972) An Experiment on Program Development. *BIT* (12): 347-365.

Schön, D. A., 1983. *The Reflective Practitioner. How Professionals Think in Action*. Basic Books.

Simon, H. A., (1960). *The new science of management decision*. Harper & Brothers, New York, NY.

Simon, H. A., (1972). Theories of bounded rationality. In, *Decision and Organization: A volume in honor of Jacob Marschak*, C. B. McGuire and R. Radner (eds.), North-Holland Publishing Company, Amsterdam, pp. 161-176.

Wirth, N., (1971). Program Development by Stepwise Refinement. *Communications of the ACM* (14:4): 221-227.