Don Bishop ©1997 Artville, LLC

Design Methodology and Design Practice

INTRODUCTION

For several decades, the field of design methodology has addressed design processes, epistemological considerations, and work practices in the design disciplines [2, 10, 11, 13]. There are good reasons for considering the development of software as a design discipline [7, 14]. Design methodology as a general field has developed across disciplines, primarily architecture, engineering design, and industrial design. This body of knowledge is not well known in our discipline but appears to be highly relevant. In this article, we introduce a number of design methods from other design disciplines and the literature on design methodology. We hope that they will prove useful for software designers.

Jonas Löwgren
Art and Communication
Malmö University
College
205 06 Malmö, Sweden
Jonas.Lowgren@kk.mah.se

Erik Stolterman
Department of
Informatics
Umeå University
01 87 Umeå, Sweden
erik@informatik.umu.se

First, a word on our view of methods. Some argue that the way to get better systems is to concentrate on developing and disseminating better methods. A good example is Butler [1], who summarizes 10 years of development in usability engineering (UE) as follows: "UE provides systematic tools and methods for the complex task of designing user interfaces that can be readily comprehended, quickly learned and reliably operated." To us, it seems clear that the result of any process will never be better than the people who participate in the process. The implication for software design is that the skills and abilities of the designer determine the quality of the final system. It follows that methods should be seen as tools for developing the designer's abilities.

By describing a particular work practice that has proved useful to other designers, we provide the reader with opportunities to develop his or her own practice. The choice of a particular method can never be made in a general way; instead, it must always be related to the situation at hand and the people involved.

John Christopher Jones published the first edition of his milestone design book *Design Methods: Seeds of Human Futures* in 1970 [4]. In the introduction, he writes about different views on the designer and on design methods. If the designer is seen as a black box, generating creative solutions without being able to explain or illustrate how the solutions came about, then the methods will be focused on facilitating and supporting the inexplicable creative processes. The other extreme is to view the designer as a glass box where every step in the design process is rational and eligible for description and transfer. Glass box methods tend to be systematic and assume sequential processes with hierarchical decomposition of problems into subproblems. A third view is to think of the designer as a self-organizing system with the abilities to search for ideas and solutions, combined with the assessment of its own processes. The corresponding methods have strong traits of meta-thinking and support the designer's reflection on work processes and design strategies.

In the terms of Jones's taxonomy, we would adhere to the view of designers as self-organizing systems with constructive as well as reflective skills. To describe a method is a way to provide designers with access to a way of working that they perhaps did not know before. The designer must assume the full responsibility for assessing the applicability and effects of the method in question, assimilate it with his current "toolbox" and use it independently and creatively in appropriate situations. Hence, we provide rather brief descriptions of the methods used to facilitate initial assessment and full references for the further studies that are needed for assimilation.

Since our purpose here is to argue that software design could benefit from other, more traditional design disciplines, we have chosen to be brief in presenting our methods. Instead, we close with a discussion on how we believe designers should relate to methods when the purpose is not only to produce results effectively and efficiently, but also to develop and improve their design ability.

Examples of the Methods

The three methods we present here are all oriented toward early phases of the design process, where concepts and ideas are the main currencies. The reason is simply that the later phases—including detailed design, implementation, and evaluation—are already addressed more extensively in the literature and are perhaps also less problematic. Our choice of methods is not meant to be comprehensive, perhaps not even the best possible. Instead we present examples of methods that can be tried out very simply without putting too much effort into the learning process. The three methods are (1) function analysis, (2) why-why-why, and (3) innovation by boundary shifting. The sidebar gives brief introductions to another handful of methods with references.

Function Analysis

Participatory design, contextual design, and many other software design methods are fundamentally based on knowing the intended

users and use contexts of the system to be developed. When a product is planned for a broad market, it is often much harder to find intended users and involve them in the design process. It is, of course, not impossible, as examples such as Kyng [5] demonstrate, but still not always feasible. The foundation on which to develop concepts is instead often a compilation of information from many different sources: studies of competing products, market research by ourselves, market research by others, interviews with people from the target population, interviews with application domain specialists, field studies of intended use situations, and so on. Landqvist [6] describes the method of *function analysis* from the field of industrial design. Its purpose is to summarize and structure the available information and to decide where more information is needed.

The main idea of a function analysis is to express *what* the future product should do (i.e., what functions it should have), but not how. Functions are expressed in two words each: a verb and a noun. There are four types of functions. There is always one *main function*—the essential idea or purpose of the product. Other functions that are required for the product to fulfill its purpose are called *necessary*. Functions that are nice to have but not necessary are *desirable*. Finally, functions that have been proposed but later ruled out are classified as *unnecessary*. Unnecessary functions are not taken out of the analysis, because they serve as a record of what went on earlier and may facilitate inter-project learning.

To illustrate a simple function analysis, imagine that we have initiated discussions with a local hospital where the administration of X-ray images appears to be a source of problems. The existing image archive is similar to a library where all the images are kept. Doctors and other authorized staff fill out request forms and the images arrive in a day or two through in-house mail, unless someone else has borrowed them.

We put together a simple function analysis in order to structure the problem and to learn what we need to find out more about. The basis for our analysis is a handful of short conversations with people at the hospital, and a demo of a commercial teleradiology product. The transcript in Figure 1 shows our first take on the situation.

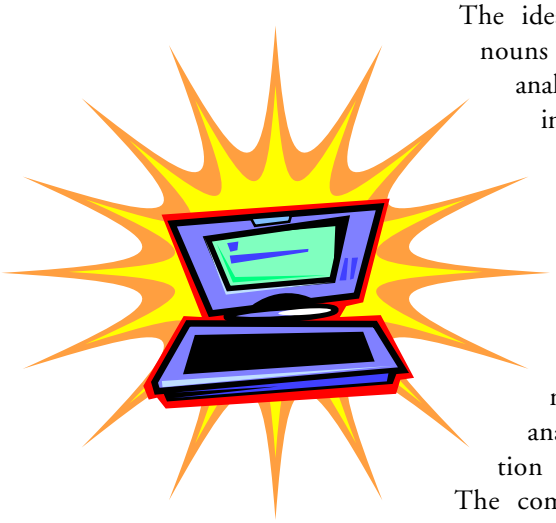
Even though the example is simple, the idea of concentrating on the "whats" of the planned system should be clear.

Landqvist [6] further recommends that the function analysis be combined with the results from subsequent how-oriented activities. For instance, if a brainstorming session (see the sidebar) is used to generate a number of design concepts and product ideas, a matrix can be constructed in which the functions from the function analysis are placed in the rows and the different design concepts in the columns. Each cell in the matrix corresponds to an assessment of how well the design concept in the column responds to the functional need in the row. If the assessments are made numerically, each column can be added up and a quantitative ranking of the design concepts is obtained; however, it might be difficult to assign numbers to complex and context-

Figure 1. Transcript from a preliminary function analysis.

Functions	Classification	Notes
Provide X-ray images	Main function	The essential purpose of the product
Search for X-ray images	Necessary	
Enhance X-ray images	Desirable	Edge detection, contrast, brightness, etc; may be permanent operations or different for each session
Store new X-ray images	Necessary	
Manage image archive	Necessary	

dependent assessments. The main benefit of the combination matrix may well lie in the systematic cross-check of all concepts against all functions.



The idea of using verbs and nouns as drivers of early analysis work can be found in many object-oriented analysis methods, where they typically correspond to methods and objects in subsequent system design. The classification of functions is a standard technique in requirement analysis within information systems development. The combination matrix suggested by Landqvist is similar to the QFD A-1 matrix that, for example, Lundell and Williams [9] report using in the same way. The main reason for presenting function analysis here is Landqvist's experience that it helps creative and concept-oriented industrial designers spend some time on the whats before diving into the hows. We believe that it can yield the same benefits in certain software design contexts.

Why-Why-Why?

It is nearly always good design practice to look beyond the problem as stated. In early phases, it is a way to direct the attention to areas where potentially powerful design concepts might be found, before any commitments are made. One way to do this is to ask a series of why questions and build a chain of connections backwards from the initial formulation [4]. To illustrate how this works, assume that the X-ray image project we mentioned earlier originated from a doctor's dissatisfaction with the current means for obtaining specific images. We create a hypothetical conversation as follows.

— *I am not happy with the way X-ray images are handled today.*
— *Why?*
— *Because I have to request images several days before I need them.*
— *Why?*
— *There is no way to know in advance how long it is going to take to get the images.*
— *Why?*
— *If somebody else has borrowed the images I want, then I have to wait until they are returned.*
— *Why?*
— *There is only one copy of each image.*

Note that the why-chains can stretch in many different directions, depending on how the whys are interpreted and whose perspective is assumed in providing answers.

— *I am not happy with the way X-ray images are handled today.*
— *Why?*
— *Because I get the wrong pictures sometimes.*
— *Why?*
— *The people at the image archive make mistakes.*
— *Why?*
— *Don't know. I suppose they are overworked.*

Why-chains like these can be used to spawn different design concepts, depending on which part of the chain is used for entry and which values are brought to the design work. The preceding examples could motivate a focus on improving the conditions in the image archive: improved procedures, rapid feedback on image requests, different modes of communication between clients and archive, several copies of each image, and so on. For a software designer, it might be natural to envision a support system for the image archive staff, dealing with digital request forms and information about the images, such as where they are stored in the archive and who has checked them out. Such a system could even notify the client automatically if all copies of the requested image are on loan. The result would be a bit like a library system.

It is important to note that the why-why-why method is merely a way of broadening the design possibilities. Formulating fruitful why-chains requires the ability to distinguish promising paths from dead ends.

— *I am not happy with the way X-ray images are handled today.*
— *Why?*
— *Because I am in a terrible mood.*
— *Why?*
— *Because my spouse and I had a fight before I left home this morning.*

The why-chains are simpler versions of the means-ends hierarchies elaborated by Rasmussen and associates in the method called ecological interface design (see, e.g., [12]). They also correspond to common design sense: do not only try to solve the problem, but also question it. The reason for elevating why-why-why to method status is that it can serve as a tool in learning the healthy practice of traversing abstraction levels in design.

Innovation by Boundary Shifting

One of the most critical stages in the design process is when the fuzzy and incomplete design concepts have to start being refined and elaborated into more tangible representations. The danger of getting locked into a specific way of regarding the problem is imminent as the time for hard commitments comes up. Many important design decisions are made, sometimes implicitly and without reflection. Jones [4] describes *innovation by boundary shifting* as a way to move the exploration outside the problem boundaries that are implicitly taken for granted and to apply knowledge from new fields to the problem. The method has four steps; we illustrate them using the X-ray image example.

1. Identify the essential functions of any device that would achieve the desired objective.

The desirable objective in our case is to improve the handling of X-ray images to remove the doctors' and other clients' dissatisfaction and problems. The essential function can be described as giving the clients access to the images they need, when they need them.

2. Identify conflicts between existing means of achieving these functions within the assumed problem boundaries.

The foremost conflict is that we cannot guarantee delivery of the proper images, even if the image archive is given a digital support system. A robot archive is not economically feasible.

3. Identify resources outside the assumed problem boundaries that might be made available by transforming the problem.

There are, of course, many potentially useful resources. Three of the most promising ones are that (a) doctors and other clients recognize the proper images when they see them, (b) clients spend time today filling out request forms and that time could be used for something else, and (c) there are good routines in place for managing other patient information in the form of journals.

4. Seek compatible solutions that would provide channels for the use of some or all of these resources.

A possible solution using resource 3(c) would be to dismantle the central image archive and keep X-ray images in the patient journals. Resources 3(a) and 3(b) can be combined in a different solution where clients use the time they spend today filling out request forms to do their own searching in a database of digitized X-ray images.

The example illustrates how the method can be used to shift the problem boundaries that were previously taken for granted; that is, that the role of the image archive is to manage and provide photographic copies of X-ray images. The possible solutions move in different directions; the first is oriented toward organizational intervention, whereas the second proposes a new storage medium for the images with a host of new implications.

We can also note that the consequences of the two ideas are quite far-reaching, even to the point that the whole design concept must be rethought. We are no longer thinking about a support system for the image archive, but rather a change in the larger system in which image providers (archive) and image consumers (clients) both take part.

Innovation by boundary shifting is a way to support "thinking outside the box" and to draw on knowledge from other fields and other perspectives when faced with a hard design problem. In this aim, it is clearly related to de Bono's notion of "lateral thinking" and the methods that are based thereupon (see [3]

Methods & Tools
Column Editors
Michael Muller
Lotus Development Corp.
55 Cambridge Parkway
Cambridge, MA 02142
+1-617-693-4235
fax: +1-617-693-1407
mullerm@acm.org

Finn Kensing
Computer Science
Roskilde University
P.O. Box 260
DK-4000 Roskilde
Denmark
+45-4675-7781-2548
fax: +45-4674-3072
kensing@dat.ruc.dk

and the sidebar for examples).

All three methods can be used as learning tools, maybe even without trying them out. It is possible to read them just to be able to formulate questions on our current practice. For instance, in what way do I approach situations described in the methods? And what are the pros and cons of my own thinking—structured or not—compared with the methods?

Design and Method

What is the role of methods in design? As stated in the introduction, the answer depends on how we regard the designers themselves. Of course, situations occur when a systematic and structured approach can relieve a designer from time-consuming and difficult tasks. This is especially the case when the purpose is straightforward production of already designed functions or systems. But in early design phases and with the view of the designer as a self-organizing system we want to advocate the idea of methods as primarily learning tools. The learning that takes place can help a designer in different ways.

By learning a new method, you extend your language and your repertoire of tools for different design situations. Even more powerful is learning a method to the level where you can go beyond the method as stated. This requires understanding why the different steps in the method are performed, adapting the method to the situation at hand, and exchanging a technique prescribed by the method for another one yielding a better result. At that level of method use, methods need not be confining or overly prescriptive. This means that what the use of methods really contribute to is the development of the designer's design ability and skills.

Another reason for using design methods, and primarily systematic methods, is that design work is always carried out in a social context. The method can serve as a common ground for more successful communication between the stakeholders in a design process. Support for coordination and planning are also benefits stemming from the social use of

More Method Examples

FUTURE WORKSHOPS can be used in short amounts of time when the intended users and stakeholders need to clarify common problems in their current situation, create visions of what future work could be like, and discuss how to realize the visions. Participants switch between small-group and whole-group work in going through phases of critique, imagination, and implementation. An example of future workshops in software design is Kensing , F., and Madsen, K. (Generating Visions: Future Workshops and Metaphorical Design. In Greenbaum, J., Kyng, M. (eds). *Design at Work: Cooperative Design of Computer Systems*, Lawrence Erlbaum, Hillsdale, NJ, 1991, 155–168).

BRAINSTORMING is probably the best known example of a creative design method and unfortunately also one of the most misused words in professional settings. Its proper use is a three-step method whereby a group of participants is selected, ideas are generated without criticism or analysis, and the results are structured to make them available for subsequent use. METHOD 635 is a variation in which six participants get well acquainted with the problem and each write down three rough ideas for solutions. The ideas are passed on to the next participant, who revises, extends, and modifies them. After five rounds of passing notes, all participants have worked on all the ideas. Brainstorming is well described in Jones, J. C. (*Design Methods*. 2nd ed. Van Nostrand Reinhold, New York, 1992). Method 635 is described by Pahl and Beitz, who attribute it to Rohrbach (Pahl, G., and Beitz, W. *Engineering Design: A Systematic Approach*. The Design Council, London, 1988).

Edward de Bono has presented many methods concerning general creativity and productivity. Some of them seem clearly relevant for software design as well. A fairly well-known example is THE SIX THINKING HATS. The idea is to make explicit different perspectives that are needed in a design process and to create more unambiguous communication. The white hat is neutral and focused on information and data. The red hat is about feelings and intuition. The black hat is for critical assessment. The yellow is optimistic and positive, and the green is for creativity and growth. The blue hat is the process facilitator.

methods. But it is important to remember that this purpose may easily become counterproductive in early design phases if creativity and boundary expansion are in focus.

Methods are bearers of historical knowledge and professional skill. Even if you do not follow a particular method to the letter in your own work, studying it can give valuable knowledge on how other designers formulate their experiences. To read and reflect upon a method description—and most of all to reinvent the underlying rationale—is a way to challenge and develop your own views and assumptions on design. This is best done if you try the method at hand in a serious way. It is in the struggle with the complexity of design situations that the real value of a method can be reflected on. When this happens, your assumptions are challenged and have to be re-evaluated and re-formulated. Donald Schön [13] presents the idea of a "reflective practicum" as an approach to reach this situation.

As a consequence, the main strength of design methods as discussed here is not in their contribution to quality assurance or repeatability. The idea of detailed method prescriptions leading to a repeatable, measurable and in some sense objective design processes is sometimes advocated in our discipline. In our view, the quality of the work in any design process can never be better than the designers and the design context, irrespective of which method is used. The case studies in Löwgren [7] provide examples of this.

A method may appear comfortable and secure, and perhaps seem to take part of the responsibility off your shoulders. Unfortunately, this security is not guaranteed. Sooner or later, you will find yourself in a situation in which the method is inadequate. A better mindset is to aim for being always *prepared to act*, by constantly developing your creative and analytical skills, assessment skills and values, rationality and communicativity, expressive and compositional skills, and knowledge of technology and of use situations. In this ongoing process, design methods are wielded as one type of tools among many others.

Another suggestion from de Bono is the RANDOM INPUT method. The problem, or creative focus, where new ideas are needed is simply juxtaposed with a randomly selected word. The new associations that emerge from the surprising combination can lead to new and useful ideas. Both the six thinking hats and random input are described in de Bono, E. (S*erious Creativity: Using the Power of Lateral Thinking To Create New Ideas. Fontana,* London, 1993).

ARGUMENTATIVE TECHNIQUES were originally conceived as consequences of viewing design as a negotiation, where there are no right answers but rather a number of more or less valuable solutions supported by more or less well-founded arguments. The issue-based information system (IBIS) notation is based on issues, positions, and arguments. Its aim was to make design more democratic by explicating and documenting the negotiations behind design decisions. Horst Rittel was the main proponent of this view of design and the originator of IBIS in the 1960s. The hypertext adaptation gIbis is described in Conklin, J., and Begeman, M. (gIbis: A Hypertext Tool for Policy Discussion. *ACM Transactions on Office Information Systems* 6,4: 303–331).

More recent examples of argumentative techniques have largely downplayed the ideological aspects of argumentative design to focus instead on documenting the exploration of different ideas and solution alternatives during a design process. The questions, options, criteria (QOC) notation is a well-known example, whereby design decisions are described in terms of Questions, Options for answering the questions, and Criteria by which to assess all the proposed options. QOC clusters can be joined as options spawn new questions, and the resulting network becomes a map of the design space traversed in the design process. The notation and underlying ideas are introduced in Maclean, A. et al. [Questions, Options, and Criteria: Elements of Design Space Analysis. *Human-Computer Interaction 6*, 3–4 (1991): 201–250].

Acknowledgments

The article is based on a chapter from Löwgren and Stolterman [8]. The X-ray image example is purely fictitious and used only for purposes of illustration.

References

1. Butler, K. Usability Engineering Turns 10. *interactions*, iii.1, (January 1996), 59–75.

2. Cross, N. *Developments in Design Methodology*. Wiley, Chichester, UK, 1984.

3. de Bono, E. *Serious Creativity: Using the Power of Lateral Thinking To Create New Ideas*. Fontana, London, 1993.

4. Jones, J. C. *Design Methods*. 2nd ed. Van Nostrand Reinhold, New York, 1992.

5. Kyng, M. Scandinavian Design: Users in Product Development. In *Proceedings of CHI'94: Human Factors in Computing Systems*. ACM Press, New York, 1994, 3–9.

6. Landqvist, J. Vilda idéer och djuplodande analys: *Om designmetodikens grunder. (Crazy Ideas and Penetrating Analysis: On the Foundations of Design Methodology.)* Carlsson, Stockholm, 1994. In Swedish.

7. Löwgren, J. Design for Use Quality in Professional Software Development. In *Proceedings of the Second Conference of the European Academy of Design* (Stockholm, April 1997). See http://www.svid.se/ead-programme.htm.

8. Löwgren, J., Stolterman, E. *Design av informationsteknologi: Materialet utan egenskaper. (Designing Information Technology: The Material without Properties.)* Studentlitteratur, Lund, Sweden, 1998. In Swedish.

9. Lundell, J., Williams, D. Integrating QFD into Software Development: A Case Study. In *Proceedings of the Fifth International Conference on Human-Computer Interaction* (HCI International '93), North-Holland, Amsterdam, 404–409.

10. Margolin, V., ed. *Design Discourse: History, Theory, Criticism*. University of Chicago Press, Chicago, 1989.

11. Margolin, V., Buchanan, R., eds. *The Idea of Design*. MIT Press, Cambridge, MA, 1995.

12. Rasmussen, J., Goodstein, L. Information Technology and Work. In Helander, M., (ed.) *Handbook of Human-Computer Interaction*, Elsevier, Amsterdam, 175–201.

13. Schön, D. *Educating the Reflective Practitioner*. Jossey-Bass Publishers, San Francisco, 1987.

14. Winograd, T., Bennett, J., De Young, L., and Hartfield, B., eds. *Bringing Design to Software*. ACM Press, New York, 1996.