THE ENGINEERING DESIGN PROCESS - DIAGNOSING THE PROBLEM: “TAKE TIME TO DEFINE” WITH CONTEXTUAL INQUIRY

Scott A.C. Flemming and Clifton R. Johnston
Dalhousie University
scott.flemming@dal.ca ; clifton.johnston@dal.ca

Abstract – In previous work [1] it was argued that turning attention from the act of problem solving to the act of problem defining has several benefits for engineering students and practitioners alike. Such benefits include developing and refining student’s empathy and critical thinking skills inside the engineering discipline. The practical question then becomes how to teach and practice the process of defining the problem well. The following paper is a literature review of the current state of problem definition within engineering design research and how problem diagnosis can be taught and practiced by engineers. Two significant insights emerge from this review: (1) traditional engineering design literature does not suggest processes for defining problems; and (2) the authors consider “Contextual Inquiry” the most promising tool for problem diagnosis from an Industrial Engineering perspective.

Keywords: problem definition, engineering design process, problem generation, problem expansion, contextual design, requirements engineering, requirements elicitation, contextual inquiry

1. INTRODUCTION

In a previous work [1] the authors explored the idea of engineering culture and it was found that such a culture results in producing engineers who struggle with self- and political awareness, understanding the worldviews of others, and hearing voices from the margins [1]. It was argued that the root cause of these negative traits was that the dominant image of engineering culture is the engineer as Problem Solver [1]. The issue is, as as one colleague has put it, engineers are taught to be “exercise completers” instead of true solvers of real-world problems [2]. An approach to building a greater awareness and understanding of others is to begin to adjust the image of the engineer to be both a problem solver and “definer” [1]. As we will see in this companion paper, defining the problem well also has other benefits: for instance, saving significant resources. We will also see that it is difficult to find processes in traditional engineering literature to aid us in discovering the problem. The authors hope that this paper will aid the reader in understanding the complexity and challenge in defining problems well and some of the techniques available to the practitioner.

This paper will focus on what fruitful methods are available to help designers in formulating problem definitions; however, one reviewer asked a key question: what characterises a well-defined problem? The short answer is that this is an under-developed field and that question would also be an appropriate subject for a literature review of its own. In the absence of such a review the authors would tentatively state (leaning on their teaching and designing experience and research thus far) that a good problem definition is a statement or set of statements that define well the system (context) in which the problem lies, the key stakeholders that this problem effects, and precisely why the problem is indeed a problem for the stakeholders (i.e. understand their worldview and work life). If practitioners can create a “How Might We” map as described by Basadur in [3], it is likely they understand the problem well. From an engineering design process point of view, a good problem statement is one that would greatly aid the requirements phase of the engineering design process. A problem is well-defined if it can provide significant help in generating a comprehensive set of requirements that would be deemed acceptable by the key set of project stakeholders. If engineers do not sense they have a well-defined problem as described above, they should be wary of moving further in the process (see Figure 1). The pitfalls of poor problem description are described in Section 2.

The rest of this paper will discuss the first step of the engineering design process and how to perform problem definition well. As the reader will notice, almost all of the techniques that are described in this paper are in the field of software engineering. This is not intentional; the authors have searched extensively for any literature pertaining to the subject of problem definition and no other discipline was found that engages the subject in a systematic fashion.

2. THE PROBLEM DEFINITION PHASE

A dozen or so textbooks were surveyed to better understand the overall design process but with specific attention paid to the nature and importance of the first step. Each text gave an overview of the engineering design process that typically included some version of the
pentagon given in Figure 1. Stage 1 was given various names such as: “task clarification” [4], “understanding the problem” [5], or “establishing the need” [6]. In several instances the importance of defining the problem well is stated with significance. For example, Hurst claims, “…much time, effort, and money can be wasted providing a solution to the wrong problem” [7]. Similarly, Ford and Coulton claim the high cost (in terms of time and finances) of re-design makes the problem-definition stage of key importance [8]. Ullman posits that “poor problem definition is a factor in 80% of all time-to-market delays” and tells a story of NASA spending millions of dollars for dampeners on the Mariner IV satellite which in the end were not required! [5] It is the position of the authors that such problems are born from designers prematurely skipping from Stage 0 to Stage 2 or Stage 3. The temptation to skip seems to be related to the elements of engineering culture as described above: students and practitioners have the tendency to see themselves as problem-solvers not as problem-“definers.” Engineers seem to feel that they are “wasting time” if they are not immediately getting tangible results. Skipping straight to say Stage 3 or 4 (or even Stage 2), however, if not done intentionally in order to learn more about the problem, can cause significant costs and delays.

![Figure 1 – Simplified Engineering Design Process](image)

The survey revealed that problem definition is not only important but also difficult: the true goals of the client or the project are often not explicitly stated [4] [9] [10] [11]; client problems are rarely crafted in “exam paper precision” [12], and initial problem statements given by clients can, and often do, “contain errors, show biases, or imply solutions” which needlessly restrict the design space [13]. Ullman puts it bluntly: “All design problems are poorly defined” [5]. If the design problem includes many stakeholders then the complexity mounts. Engineering projects in the sustainable development field are an extreme example of the complications that accompany multiple stakeholders with widely-varying worldviews and goals. Lucena reports that if a project is not defined well in terms of all the relevant stakeholders then the undertaking will suffer with respect to “ownership, success, and long-term sustainability” [14]. Lucena et al. describe an unsettling example of engineers almost installing toilets for clients who wanted an windmill. See Chapter 6 in [14].

Authors from other (related) fields also describe the ills of defining the problem poorly and the benefits of defining it well [3]. A famous quotation often erroneously attributed to Einstein (actually an Industrial Engineering Department head from Yale) highlights the importance of problem definition: “If I had only one hour to solve a problem, I would spend up to two-thirds of that hour in attempting to define what the problem is.” [15] John Dewey emphasises the same point when he writes, “A question well-put is half-answered; a difficulty clearly apprehended is likely to suggest its own solution,” while also affirming the danger of the opposite scenario: “while a vague and miscellaneous perception of the problem leads to groping and fumbling” [16]. Perhaps John Tukey is pointing out our anxieties to prematurely solve problems when he writes, “Far better an approximate answer to the right question, which is often vague, than an exact answer to the wrong question, which can always be made precise.” [17]. In short, this section makes plain that diagnosing the real problem is a vital task.

Defining the problem is a vital and non-trivial task. To understand the problem better one must of course speak to one’s client(s) to learn more about their problems (whether they are consumers in the mass market or particular clients in a specific consulting project). Here various authors hint at best practices in doing so. Some examples offered are that a collegial and non-competitive approach is required [13], one should attempt to probe until the “real” problem is discovered [4], engineers will have to accept that data gathering will be unsystematic at first since they are not yet certain of the exact nature of the problem [7], and “As we gather information from clients, users, and other stakeholders our views of the design problem will shift as we expose implicit assumptions and perhaps a bias toward an implied solution” [13].

The reader may be expecting at this point a summary of systematic methods for diagnosing problems. A somewhat baffling result of this literature review is that there is no such summary available from traditional engineering design sources. Although almost every engineering design textbook had something to say about problem definition, they did not have much written about how one was to go about doing it. Consider one design text subtitled “A Systematic Approach.” This particular text had only four of its 617 pages devoted to a discussion on defining.
problems [4]. The fact that there exists so little content regarding the attributes of a good problem definition or how to go about producing one gives strong evidence to support the claim that the problem diagnosis phase of the design process is overlooked and underdeveloped in traditional engineering – even though it is clearly considered to be “essential” [18].

What this literature review has produced, then, is a series of characteristics which describe good approaches to defining problems rather than processes detailing how one should attempt to define problems well. These characteristics, the authors believe, should be considered criteria for evaluating approaches to problem definition that were found from other fields and described in Section 3. In other words, a “good” approach to defining the problem should: (a) encourage collegiality with one’s client, (b) enable engineers to enter into the worldview of their clients and stakeholders, and (c) enable designers to defer or suspend their judgements about what the problem really is until it emerges on its own. A fourth practical concern (d) is that the process should not create an overwhelming workload for designers. These criteria are explained in more depth below.

A good approach to defining a problem should (a) encourage a collegial interaction with one’s client. One could argue that any particular approach to problem diagnosis could be deemed collegial or not depending on the designer and how they use it. To be more precise, the authors mean an ideal problem-defining tool would be able to develop the skill of collegiality, perhaps even compassion, in the designer. The authors would like to point out that oftentimes engineers work with a particular and finite number of clients. This means that while some approaches may be meant for learning about many customers as a whole, for engineering design purposes often this is not enough. In many contexts engineers must learn about their particular client and customer needs which may be counted in the tens instead of tens of thousands. In these cases, collegiality is all that much more important.

The second requirement for a problem-defining method is that it should (b) enable designers to enter into the worldview of the users or customers. Here again, one could argue that many methods of customer discovery or analysis can help a designer understand facts about or data regarding their client. We suggest, however, that an excellent method would have the ability, if followed well, to open the eyes of the designer so that they may see problems through the eyes of their users – not just understand more about clients in general. Ideally the designer would come to an “ahah!” moment realizing that, “Now, I understand why they were having such difficulty!”

Thirdly the authors suggest that a good method of defining a problem would (c) aid the designer in avoiding prematurely diagnosing the problem in their context. The output of the problem-discovery process should be some kind of outline of client experience and the problems with their experience. While such an outline is being created there ought to be recognizable markers for determining when one discovers something of potential value. In addition, to minimize frustration of the designers there should be an indication as to when the problem definition phase can be ended (at least temporarily). A superior approach will be one that indicates to the designer when “enough is enough” and the next phase of the design process can begin.

Lastly and fourthly, the workload (d) of these various methods is an important criterion as some of the techniques require a significant investment of time and resources.

The next section will be devoted to describing the available literature that can help designers better understand how to move from Stage 0 to Stage 1 and resist the temptation to jump to Stage 2 or Stage 3 as described in Figure 1. In their research the authors discovered that the area called “Requirements Engineering” (RE), a sub-discipline of Software Engineering, concerns itself with exactly this part of the design process: defining the problem well in terms of goals, specifications, and constraints [19]. The authors surmise from what they have read that eliciting the precise goals and specifications for a software product is particularly important as, if the engineer is not careful, the product delivered can be very different than the product the client expects – much more than in most other fields. Since setting expectations is so important in this field a great host of techniques has been detailed in how one goes about translating client needs and problems into well-defined language. Nuseibeh and Easterbrook have created a very clear and concise categorization scheme for this host of problem-diagnosing (“requirement elicitation” in their parlance) techniques: prototyping, traditional, group, model-driven, cognitive, and contextual [19] [20] [21] [22] [23].

3. EXPLANATION AND EVALUATION OF ELICITATION TECHNIQUES

This section is meant to give a high-level view or understanding of problem-definition tools for educators and practitioners that wish to expand the problem definition stage of their design process. Table 1 - Evaluating Methods of Problem Definition illustrates which of the techniques from software engineering would be most valuable to more traditional engineering disciplines. The authors would like to be clear that they have evaluated the following approaches from the particular context of teaching and practicing engineering design in Industrial Engineering (IE). Each proposed method below then was evaluated from that perspective and according to the four criteria outlined in detail in Section 2. Each scheme for problem definition was given a score of unsatisfactory (x), acceptable (·), or very good (✓) with a discussion of those results to follow. If a
particular technique is given more than one symbol for how it scores on a criterion, this specifies that “it depends.” For example, “traditional” methods of problem definition have been scored as “✓, ~, ✗” with regards to workload as the level of effort depends on how the methods are executed.

Table 1 communicates that from an IE perspective the authors believe Group, Model Development, and Contextual methods are most promising in aiding the problem definition phase of the design process. It is our opinion, after this literature review, that the single most promising method that will help practitioners define problems and instructors teach problem definition in IE is the Contextual Method. A more in-depth analysis follows below.

| Table 1 - Evaluating Methods of Problem Definition |
|-----------------------------------------------|
| Method   | Collegiality | World-view | Discovery | Workload |
| Prototyping | ~            | ~           | ~         | ✓, ~, ✓  |
| Traditional | ~            | ~           | ~         | ✓, ~, ✓  |
| Group     | ✓            | ✓           | ✓         | x        |
| Model-Dev | ✓            | ✓           | ✓         | x        |
| Contextual| ✓            | ✓           | ✓         | ~, ✓     |
| Cognitive |             |             |           | Not applicable |

3.1 Prototyping

Prototyping normally occurs later in the design process but can be used in this first stage in order to identify customer needs and issues depending on the context [20]. In this early phase low fidelity prototypes can be especially useful in helping stakeholders give meaningful feedback and refinements [20]. Of course, just because an engineer makes a prototype does not mean that he or she is open to client feedback about it. However, prototypes are useful methods of gaining insight into the real nature of the problem and give the users an “active role in developing the [project] requirements” [20]. This method is especially useful if clients are not aware of the possible solutions that can exist to solve their problem [20] and/or there is significant uncertainty about what the requirements for the project really are [21]. Since prototyping in-and-of itself does not guide the engineer toward collegiality, understanding of client worldview, or the art of discovery as much as some of the other methods it was evaluated as only acceptable (~) across the criteria (see Table 1). This is not to say that prototyping is not valuable or even the best method in some circumstances; it is only to say that in our typical contexts as Industrial Engineers other methods appear to be more promising.

3.2 Traditional Data Gathering Methods

“Traditional” method of requirements elicitation are those methods of data gathering that are quite familiar to most engineers such as questionnaires, surveys, question-and-answer interviews, analysis of documentation, etc. [11] which overlap significantly with methods common in commerce [21] [24]. These methods typically focus on quantitative data which can be misleading especially in the early stages of design. With this approach the customer seems to be more of “a quantity to be measured” (by using surveys, for example) for the sake of developing requirements more than “a partner to be worked with” in order to diagnose essential problems [5]. In these approaches much work is done to attempt to understand who the customer is and what they want, but it is done mainly at arm’s length through research, literature reviews, and benchmarking [11] [24]1. While of course not explicitly anti-collegial, these approaches do not encourage collegiality with the customer and often assume the customer is always a part of some very large consumer group. Architectural Programming assumes a similar posture in learning more about their clients [25]. These sorts of techniques do take the problem definition stage of their work seriously and aim to include and understand a multiplicity of stakeholders; but, they seem to devolve into a structured process of simply amassing as much data about as many stakeholders as possible under various categories. While admirable and not adversarial, the authors do not believe that these approaches could be described as particularly collegial as defined in Section 2. These approaches have been evaluated as acceptable (~) since they do describe the importance of getting buy-in from different stakeholders, garnering consensus, and so on but do not give many particular suggestions concerning how one should proceed in executing those activities [25]. Similarly, the traditional methods of data gathering do not prevent one from understanding the worldview of the client or discovering something of rich value but seem to fall short in guiding the designer in producing deep insight.

3.3 Group Elicitation Techniques

Group techniques attempt to facilitate consensus for the various stakeholders in a given project as well as make the goal of the project very clear [21]. The title “Group Techniques” casts a wide net: this includes brainstorming and various methodical variations on brainstorming such methods as described. CI will be further discussed in Section 3.6.

---

1 A notable exception is MacQuarrie [31] who uses Contextual Inquiry (CI) in conjunction with traditional
as Joint Application Development, Requirements Workshops, and Rapid Application Development [20][21][23]. These methods have the objective of consensus-building and discovery of the needs/desires of key stakeholders [20]. They differ in that some are more structured than others and some have slightly different purposes. Many if not all of these require a greatly skilled facilitator to manage politicially challenging situations gracefully and ensure that minority voices are heard [20]. The authors have scored this rather large category as performing well in all the criteria mentioned in Table 1. Such tactics, delivered by a skilled facilitator, seem very promising in promoting a collegial interaction, an understanding of worldviews of participants, and a structured approach that guides discovery. The downside mentioned several times in the literature is the human capital required to run such meetings effectively [19][20][21].

### 3.4 Model Development Techniques

Goal-modelling techniques document goals, sub-goals, and various domain entities [20][26]. They use AND/OR logic and attempt to describe the goals of a system in conceptual models instead of prose [26]. Various specific techniques can be used such as GBARM, GDC, KAOS, I#, and NFR [20][26]. Such techniques are especially helpful if only the high-level system needs are known and details of specific issues are not well understood [20].

Another modelling method is the Viewpoints Modelling technique. This is where the designer attempts to view the system from various angles – whether it be from the point of view of various users or with the goal of seeing how one aspect of the system is affected by all the others (e.g. focussing on the operations or implementation of the system) [20]. There are various particular methods for engaging in a viewpoint analysis such as PREview and CORE one can choose from. The system that PREview uses, for instance, is to name the viewpoint, determine the focus, elaborate the viewpoint’s concerns, and list the data source, requirements, and history [27]. There is also guidance in which viewpoints to choose and when to stop collecting data [27].

These methods are rigorous and have great promise for exposing key requirements for engineering projects and meet the first three criteria well. These methods are, however, very time-intensive and the payback may not be sufficient for the client in traditional engineering design challenges unless the practitioners are entering into a relatively unknown area.

### 3.5 Cognitive Elicitation Techniques

Cognitive techniques, such as laddering [28] were derived from the field of Knowledge Acquisition and can be used for complex knowledge-based systems [21]. Since these knowledge systems are almost exclusively created and managed in the field of Computer Engineering we will not discuss them here. The authors believe that such techniques would be of limited use for other engineering disciplines. Some aspects of cognitive techniques that may be helpful (e.g. protocol analysis) can also be considered as belonging to the next group: Contextual Techniques [21]. Other heuristics or tricks such as the “5 whys” are rather simple and can help in small problems but are not significant enough on their own to be called an elicitation technique – useful as they may be in some scenarios [29].

#### 3.6 Contextual Elicitation Techniques

The idea behind Contextual Techniques is to take the stakeholders’ tasks, domain, and physical work location into serious consideration when understanding the nature of the problem and client needs [21]. Various methods fall under this large umbrella such as: task analysis, domain analysis, ethnography, observation, protocol analysis, and apprenticing [20]. Contextual Design (CD) is one such procedure, detailing how designers should observe, interview, and learn from their stakeholders. The goal is to understand the nature of their work very well so that they can design tools to help them be more effective in their work [30].

The contextual (sometimes called ethnomorphological) approaches lend themselves well to fostering collegial interactions with clients. In the Contextual Design method in particular, for instance, engineers are asked to volunteer to conform to the Master-Apprentice model (with their being the Apprentice) when interacting with the client: they are encouraged to always “[t]ake the attitude that nothing any person does is done for no reason” [30]. In this method, the engineer-designer is coached to understand that the client has much to offer; and, although the engineer brings crucial technical skills to the team, the client is the expert when it comes to their job and what is required to do it well [30].

Contextual methods also do very well in enabling engineers to enter into the world of their clients. Holtzblatt and Beyer, for example, guide their readers in a precise and methodical way how to enter into their client’s world [30]. They make a unique and powerful contribution in that they suggest a detailed method of how to do this while also giving engineers the freedom to follow various leads that they deem important given that they are experts in their own discipline.

CD helps to suspend designers’ judgements on the true nature of the problem by giving a structured process to follow during what they call the “contextual interview.” This helps designers set expectations for themselves and the client, understand the purpose of the interview, and to know what to look out for while they are interviewing a particular client (e.g. difficult-to-interpret client actions, emotions, and workarounds) [30]. For all the reasons
stated above, Contextual Methods have scored very well in terms of the criteria introduced in Section 2. Like many methods, however, the work and resources required can be substantial depending on the scope of the study.

If the reader would like to learn more about this method the authors suggest reading Chapter 3 of Holtzblatt and Beyer’s text [30]. Their chapter describes well the general approach and process for Contextual Inquiry.

4. SUMMARY OF FINDINGS

Research was conducted for the purpose of gaining a macro view of what tools are currently available to aid in the problem diagnosis stage of engineering design. Traditional sources in Engineering Design described the positive characteristics good approaches to problem definition (encouraging a collegial approach, promoting understanding of the worldview of the client, and enabling discovery) but did not describe how to execute problem definition well. When other sources were sought, especially those in the area of Requirements Engineering (RE), several methods of how to perform problem definition were outlined in great detail. Given the authors’ expertise and experience consulting and teaching in Industrial Engineering, the most appropriate methods of problem diagnosis were deemed to be Group, Model Development, and Contextual. In particular, the authors suggest that the most promising approach is the Contextual Interview given that it has a well-defined method which encourages collegiality with the client, concrete ways of helping the engineer understand the worldview of the client, indicators of when promising areas for improvement are encountered, and a reasonable time required for completion. We hope the brief descriptions and works cited will enable faculty members from other engineering disciplines to determine which specific approaches may be most promising for their particular discipline.

5. FORTHCOMING WORK

The authors are currently planning to follow this paper and the previous paper [1] with a rapid ethnographic study of best practices for problem definition in Industrial Engineering. Practices observed will be contrasted with those described in the literature review with the intent of adding to the work in this under-served area and developing teaching materials at the undergraduate and graduate levels.

Acknowledgements

The authors would like to thank NSERC for their generous support of this research.

[9] Y. Haik, Engineering Design Process, Boston: Cengage Learning, 2018.
[10] N. Cross, Engineering Design Methods: Strategies for Product Design, Toronto: Wiley, 1994.
[11] R. J. Eggert, Engineering Design, Upper Saddle River: Pearson Prentice Hall, 2005.
[12] G. L. Glegg, The Design of Design, New York: Cambridge University Press, 1969.
[13] C. L. Dym and P. Little, Engineering Design: a Project-Based Introduction, Toronto: John Wiley, 2000.
[14] J. Lucena, J. J. Schneider and J. A. Leydens, Engineering and Sustainable Community Development, San Rafael: Morgan & Claypool, 2010.
[15] R. E. Finley and H. R. Ziobro, “The Manufacturing Manager's Skills,” in The Manufacturing Man and His Job, New York, American Management Association, 1966, pp. 15-21.
[16] J. Dewey, How We Think, New York: Dover Publications, 1997.
[17] J. Tukey, “The Future of Data Analysis,” The Annals of Mathematical Statistics, vol. 33, no. 1, pp. 1-67, 1962.
[18] S. McCall, Designing Engineers: An Introductory Text, Hoboken: Wiley, 2015.
[19] P. Zave, "Classification of research efforts in requirements engineering," *ACM Computing Surveys*, vol. 29, no. 4, pp. 315-321, 1997.

[20] D. Zowghi and C. Coulin, "Requirements Elicitation: A Survey of Techniques, Approaches, and Tools," in *Engineering and Managing Software Requirements*, Berlin, Heidelberg, Springer, 2005.

[21] B. Nuseibeh and S. Easterbrook, "Requirements engineering: a roadmap," *ICSE '00 Proceedings of the Conference on The Future of Software*, pp. 35-46, 2000.

[22] J. Goguen and C. Linde, "Techniques for Requirements Elicitation," *1st IEEE International Symposium on Requirements Engineering*, pp. 152-164, 1993.

[23] N. Maiden and G. Rugg, "ACRE: Selecting Methods for Requirements Acquisition," *Software Engineering Journal*, vol. 5, no. 4, pp. 183-192, 1996.

[24] S. Blank, *The Four Steps to the Epiphany: Successful Strategies for Products that Win*, California: K&S Ranch, 2013.

[25] W. Peña and S. Parshall, *Problem Seeking: An Architectural Programming Primer*, Hoboken: John Wiley & Sons, 2012.

[26] K. Pohl, *Requirements Engineering: Fundamentals, Principles, and Techniques*, Heidelberg: Springer, 2010.

[27] I. Sommerville, P. Sawyer and S. Viller, "Viewpoints for Requirements Elicitation: A Practical Approach," *Proceedings of the IEEE Conference on Requirements Engineering*, pp. 74-81, 1998.

[28] C. Chun-Hsien, P. K. Li and W. Yan, "A strategy for acquiring customer requirement patterns using laddering technique and ART2 neural network," *Advanced Engineering Informatics*, vol. 16, no. 3, pp. 229-240, 2002.

[29] P. S. Pande, R. P. Neuman and R. R. Cavanagh, *The Six Sigma way : How GE, Motorola, and other top companies are honing their performance*, New York: McGraw-Hill, 2000.

[30] K. Holtzblatt and H. Beyer, *Contextual Design : Design for life*, Amsterdam: Morgan Kaufmann, 2017.

[31] E. McQuarrie, *Customer Visits: Building a Better Market Focus*, New York: Taylor and Francis, 2008.