How to Solve Real-Life Problems: Lessons from Air Force Leadership

Martine Ceberio, Olga Kosheleva, and Vladik Kreinovich

Abstract In a recent book, two veteran Air Force leaders provide general advice on how to deal with real-life challenges. In this paper, we summarize this advice in precise terms, and explain that this advice fits with common sense.

1 Introduction

What we do in this paper. A recent book [2] from two veteran leaders of the US Air Force provides advice on how to deal with real-life problems. In the book, this advice is given in storytelling form, based largely on the book authors' personal experience.

In this short article, we summarize this advice in a systematic and precise way.

How this paper is structured. In this article, chapters and pages refer to [2].

The summarized advice is highlighted by italics. Non-italicized text contains our comments.

Martine Ceberio
Department of Computer Science, University of Texas at El Paso, 500 W. University
El Paso, Texas 79968, USA, e-mail: mceberio@utep.edu

Olga Kosheleva
Department of Teacher Education, University of Texas at El Paso, 500 W. University
El Paso, Texas 79968, USA, e-mail: olgak@utep.edu

Vladik Kreinovich
Department of Computer Science, University of Texas at El Paso, 500 W. University
El Paso, Texas 79968, USA, e-mail: vladik@utep.edu

2 Sometimes, we need to act urgently

Sometimes, we face an urgent situation, when something happened suddenly, and an immediate action is required. In this case, there is not much time for a thorough analysis – *we need to do what we can* (Chapter 12): we need to *build a plan and follow it* (Chapter 11, p. 103).

We cannot do much analysis when such a problem appears, but what we *can* do is prepare beforehand for such situations; see below.

Good news is that in many cases, there is time to think before acting.

3 When we face several problems, which one(s) should we focus on

In many cases, we face several real-life problems, and it is not realistic to solve all of them. In such cases, it is necessary to select a problem that we will focus on solving.

In this case, a natural advice is to *select a problem in solving which you are most likely to succeed – because it best fits your strengths and talents* (Chapter 4). To be able to do that, we need to determine which are our strengths and talents – by analyzing our prior experiences. The book calls this advice "find your purpose".

4 Once we selected a problem, first, we need to understand it better

When we face a real-life problem, first, we need to understand it better (p. 30, Section 3). We "need to *clearly articulate a problem statement* before acting on a proposed solution" (p. 80, Chapter 9).

This is similar to what we Computer Science faculty teach students in our Software Engineering classes: when the real-life problem is not formulated in precise terms – and real-life are rarely formulated precisely – they should not immediately start coding based on their understanding of the problem. Such coding is often a waste of resources – since the programmer's initial understanding of the problem is often different from what the user really needs. In our two-semester Software Engineering sequence, the students are presented with a real-life problem at the beginning of the first semester. Usually, the whole first semester is focused on the proper formalization of the problem, in dialog with the user(s). And only in the second semester, when both the students and the user(s) have a common understanding, the actual coding starts.

How can we understand the problem better:

- we need to *collect as much data* about this problem *as possible* (Chapter 3, p. 31);

- we need to *elicit and collect opinions of others*, we need to actively listen to what people say (Chapter 9, p. 82);
- we need to actively *use our own intuition* (Chapter 7) – what the book calls "trust your guts";
- finally, we need *to take into account what can be done* within given time and with other resource constraints; in real-life situations, it is usually enough to satisfy only some of the requirements (Chapter 6, p. 56).

5 Once a problem is clearly formulated, how should we actually solve it

How to make gaps narrower. Real-life problems are usually very complex; the gap between our state of knowledge to the desired solution is usually too large to overcome it in one leap. To make leaps smaller, we need to come up with some intermediate steps. There are two possibilities to do it:

- one possibility is to analyze the problem itself and come up with natural intermediate steps, so that transitions between each step and the next step will be easier; this known as *planning* (Chapter 3, p. 30);
- another possibility is to look for intermediate steps that do not naturally follow from the analysis of the problem but that would be helpful – in the sense that their solutions would help us solve our problem; in other words, we need to look for *similar problems* – with a special emphasis on similar problems that have already been fully or partially solved (Chapter 3, p. 32).

And, of course, we can try to use both strategies.

Who to collaborate with. Since real-life problems are complex, we often need additional collaborators. How to select collaborators? According to the book (Chapter 6, p. 54), the most important criterion is enthusiasm, we need to *select collaborators who are enthusiastically pro-active*.

Solve at least some aspects. If we cannot solve all the aspects of the problem – we need to *prioritise some aspects and work on them* (Chapter 3, p. 33) – in line with the previously mentioned advice to take into account that it is usually enough to satisfy only some of the constraints and requirements.

What if this does not work. If this does not work – *try unusual ways*; it sometimes helps. Sometimes it does not help right away, but the resulting experience helps later on, in solving future problems (Chapter 6, p. 57).

And it makes sense to *take risks*, to concentrate resources on a strategy that has a high probability of failure (Chapter 6, p. 56). This may not help us solve this particular problem, but this experience often helps to solve future problems. After all, to be successful, a machine learning tool needs to know, for each task – be it playing chess or doing something more practical – not only many positive example, but also an equal number of negative examples.

6 Need to thoroughly test a proposed solution

Before implementing the proposed solution in real life, it is desirable to *test the solution on simulations as much as possible* (Chapter 9, p. 78).

7 How to present a solution

- The *solution should be presented in a clear and explainable way*, “providing clear, concise, and thoughtful guidance” (Chapter 13, p. 125).
- It is also important to *emphasize the imperfection of the proposed solution* – imperfection that is inevitable since real-life problems and real-life situations have a lot of uncertainty (Chapter 13, p. 131).

8 Post-analysis is important

Solutions to real-life problems rarely work perfectly. Once we know the results of using our solution, it is important to *analyze what we could have done better* (Chapter 7).

Yes, often, the low performance – and even failure – is caused largely by factors that are not under our control, but usually, we can still understand what we could have done better.

Comment. All these steps are similar to what is known as the IDEAL approach to problem solving:

- Identify the problem – this is what we considered in the beginning of Section 4;
- Define what we want to find – this is what we mentioned in the last paragraph of Section 4;
- Explore possible solutions – this is described in the first subsection of Section 5;
- take Action – this corresponds to Section 6; and
- Look back – this corresponds to Section 7.

9 How to prepare for future problems

How can we better prepare for future problems? What helps in solving problem is:

- our own experience and
- experience of others – that is often reflected in different available tools.

So, the natural advice is:

- to boost our own experience – *when there is an opportunity* to participate in solving a problem, *take it* (Chapter 13); book names it "answer the call"; and
- to better utilize others' experience, to *collect tools* (Chapter 3), even if they are not useful right now; this is an important part of *lifelong learning*.

The advice about collecting tools is similar to how the renowned science fiction writer Stanislaw Lem – best known to the US audiences as the author of the novel "Solaris", the basis for a 2002 film – describes mathematics and its often surprising usefulness in [1]. Lem compares mathematics to a crazy tailor that makes suits for alien creatures of all shapes and sizes. At each moment of time, most suits remain unused, but once in a while, a weird-shaped creature appears, and the tailor produced a suit that is a perfect fit for this creature. In mathematics, this happened many times. For example:

- a very theoretical notion on non-Euclidean geometry, in particular, of Riemannian geometry, turned out to be a perfect fit for Einstein's ideas about gravity and space-time,
- abstract mathematical notions of imaginary and complex numbers, of matrix algebra, and of infinite-dimensional analogs of the Euclidean space turned out to be a pefect fit for quantum physics, etc.

10 What if there are no visible problems

Sometimes, in some areas, it may look like there are no real-life problems. Experience shows that such an absence of problems is an illusion, what the book calls *blind spots*. Serious problems surface later – and it is always better to start solving a problem as early as possible. From this viewpoint, in situations in which there are no visible problems, we need to *actively search problems* – e.g., by asking for people's opinions (Chapter 5).

Acknowledgments

This work was supported in part by the National Science Foundation grants 1623190 (A Model of Change for Preparing a New Generation for Professional Practice in Computer Science), HRD-1834620 and HRD-2034030 (CAHSI Includes), EAR-2225395 (Center for Collective Impact in Earthquake Science C-CIES), and by the AT&T Fellowship in Information Technology.

It was also supported by a grant from the Hungarian National Research, Development and Innovation Office (NRDI), by the Institute for Risk and Reliability, Leibniz Universitaet Hannover, Germany, and by the European Union under the project ROBOPROX (No. CZ.02.01.01/00/22 008/0004590).

References

1. S. Lem, *Summa Technologiae*, University of Minnesota Press, Minneapolis, Minnesota, USA, 2014.
2. H. Wilson and D. Goldfein, *Get Back Up: Lessons in Servant Leadership*, The University of Texas at El Paso Press, El Paso, Texas, USA, 2025.