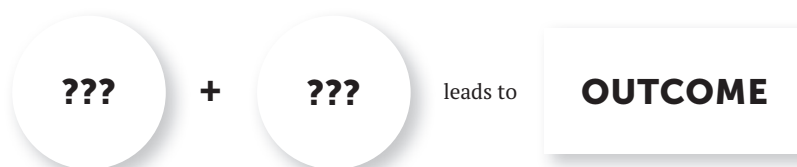


new scenarios. Normal abduction is the reasoning pattern behind conventional problem-solving—using the tried and tested patterns of relationships to reach a solution. And this should not be dismissed: often the patterns of relationships that have been developed over many years of problem-solving efforts are more than adequate to deal with the problem situation at hand. But sometimes this type of routine reasoning doesn't lead to the desired value anymore, and we will have to think about the problem again. That brings us to the second type of productive reasoning, design abduction.

*Design abduction—two unknowns lead to a process of creative exploration*



In design abduction, the starting point is that we *only* know something about the nature of the outcome, the desired value we want to achieve. So the challenge is to figure out “what” new elements to create, while there is no known or chosen “how,” a “pattern of relationships” that we can trust to lead to the desired outcome. Thus we have to create or choose both a “how” and a “pattern of relationships.” As these are quite dependent on one another, they should be developed in parallel. This double creative step requires designers to devise proposals for both the “what” and the “how,” and test them in conjunction.

An example can help to clarify the difference between the two types of abduction: say that the *outcome* we want to achieve is an energy rush when coming to work in the morning. In normal abduction, we would also already know the “how,” say that this is to be achieved through coffee—and we might even have a proposed method of brewing coffee (dripping, squeezing, using steam) so we can start developing a “what,” engineering the machine to make the coffee for us. In design abduction, on the other hand, we would only know the goal (quick rush of energy before work) but not know how to achieve it. Hence, if we go for coffee, we would still need to choose a brewing method, create a design for a machine, and then judge whether this would do the trick