Lynn Huynh

Zhaojun Bai

Great Ideas in Computer Science

13 March 2013

Spirit of Computing: Abstraction

The concept of abstraction is seemingly abstract in itself. Defined as an idea not associated with any specific instance, abstraction first requires some instantiation before one can become more familiar with this concept. In other words, to better understand what abstraction is, or to make this notion less abstract, one must fill in the details inherent to the idea itself. Abstraction can be thought of in a variety of ways but its essential purpose is to manage all the details. In abstracting something, one can remove certain details that are irrelevant to the primary focus. As such, some of those details might need to be "filled" back in if we are to think of the idea as a whole again. Yet the process of abstraction is generally applied to extremely complex situations. The need to "fill" the details back in is therefore unnecessary because the underlying rationale for abstraction in the first place, is to simplify. It can also be seen as a process of generalization in order to identify the common essence. Recognizing the core features that each component shares, enables one to abstract a complicated system into a more straightforward simplification of detail. Consequently, only the level of detail necessary at the time has to be considered. In "Algorithmics: The Spirit of Computing", David Harel brings up the analogy of baking a cake. As opposed to saying "take 2365 grains of powdered sugar, pour them into the melted chocolate, pick up a spoon, and use circular movements to stir it in", instructions such as "stir in sugar" are sufficient. Clearly, the level of detail encompassed in either set of directions is different. We can exclude many of the extremely specific commands in the first set because it is assumed that we understand what it means to carry out the general instructions. It can be simplified because we already have some understanding of this abstraction and are able to fill in the details without being explicitly told what to do. This example also highlights the importance in balancing the level of detail to that of the comprehension capability of the user. Similarly, an algorithm's instructions must be tailored to the specific hardware. As an extension of the cooking analogy, the algorithm can be thought of as the "recipe" for baking the cake. The algorithm provides instructions for operations which the computer will carry out. Different algorithms are designed according to the different levels of abstraction in the computer. Instead of thinking in terms of each "bit", or each "grain of powdered sugar" as in the cooking analogy, we use abstraction to generalize. For instance, a group of bits (with 8 bits making up a byte) can represent a character, a sequence of characters can signify a word, a sequence of words can denote a sentence and so forth. The many different levels can be drawn onto paragraphs, chapters and even books. Algorithms correspond to each of these levels of detail appropriately. An example of this can be seen in the way that the spell-check tool applies specifically to words, and not characters or paragraphs. The algorithm focuses on the relevant level of abstraction while ignoring all other details. In a sense, this enables it to communicate more effectively with the computer. In general, computers are capable of performing only very simple operations. Yet we can make a computer understand higher level abstractions by appropriately programming it. By starting off at a basic level and setting different characters to signify different ideas, we can simplify and abstract away to the different levels of detail. Therefore, abstraction may also be considered to be an instrument for acting on imaginary things that represent real objects. Overall, its role in simplifying complex processes makes it an important concept in computer programming.