Journal of Applied Mechanical Engineering
ISSN: 2168-9873

Rubin and Tebibel, J Appl Mech Eng 2016, 5:1
DOI: 10.4172/2168-9873.1000190

Commentary Open Access

The Future of Intelligent Design

Rubin SH[1] and Tebibel TB[2]*

[1]SSC-PAC San Diego, CA 92152, USA
[2]Ecole nationale Supérieure d'Informatique, BP 68M Oued-Smar, 16309, Alger, Algeria

Mechanical engineers (ME) are generally motivated to build things. The typically attend four or more years at an ABIT-accredited engineering school, learn computer technology (e.g., CAD/CAM), mathematics (e.g., matrix theory), and material science (e.g., mechanics of materials). But, just as the slide rule has been obsoleted by the calculator, we have to ask three questions to get a better perception of the field; namely, where have we been, where are we now, and where are we going (or want to go) in the future?

To answer the first question, we began as an ad hoc field – designing bridges (now the province of civil engineers), steam engines, and all manner of machines. We used imprecise methods and seat of the pants reckoning. Today, we have evolved into a practical science, which is ever-more computer-based – using everything from finite-element modeling, to elaborate stress and strain simulations, to physical, chemical, and/or mathematical analyses of strength of materials. Not that many years ago, we wrote our own Fortran codes. Today, we have evolved into using packages for everything – many of which may be found on Matlab. But, while such packages support engineering mechanics and insure integrity, they do little to support design. Sure CAD/CAM systems enable the typical ME to be an order of magnitude more productive than their predecessors, who used drafting boards. But, why is this? The answer is three-pronged. That is, the computer enables (1) reuse, (2) intelligence, and (3) abstraction.

The computer enables reuse principally through databases and case bases. Previous designs may be stored and reused – only to be modified, indexed, and reused once more. The goal is to encapsulate design decisions for reuse and indexing. Expert systems have not worked well in this area because it is very difficult to capture all the nuances involved in a design decision. It is even more difficult to propagate the changes made from one design decision to another. For this reason, case-based reasoning is used to capture designs in an episodic memory for replay.

Intelligence refers to transformation in the context of mechanical design. For example, if one doubles the weight on an axle, what is the increase in diameter necessary to compensate? Similarly, how many stresses/strains (e.g., Hookes modulus) over what period of time justifies switching from steel to titanium in such critical joints (e.g., rotor connections)? In a case-based design system, intelligence refers to a capability to generalize one design to create others by process of instantiation. For example, the case might describe the parameters for the design of a small ship hull for doing say 20 knots. How might those parameters be generalized to be applicable to a large ship hull for doing say 30 knots? The answer is that some designs can transfer to others by process of randomization, or compression. This is akin to some forms of directed data mining, where data can be replaced by equational reductions. Where this works, the solutions are said to be symmetric. Otherwise, they are said to be random, or incomparable.

Thus, if the design ME can formally apply the design principles for one design to another, the pair is said to be symmetric. Often, otherwise random designs can be brought into symmetry through stepwise transformation. This points out the advantages of having an economy of scale.

The third prong is abstraction. Higher-level programming languages (e.g., CAD/CAM based on LISP) are abstractions of their predecessors – i.e., specific Fortran codes. Abstractions allow the ME to work non-procedurally. That is, instead of telling the computer what to do, the ME tells the computer what needs to be accomplished and lets the computer program supply the details. This characterizes fourth-generation languages. The problem is that such languages are not universal and result in an explosion of special-purpose commands (e.g., similar to the methods found in the JAVA programming language).

The so-called fifth-generation languages are the fourth-generation languages supplanted with intelligent methods and procedures (e.g., PROLOGUE for logical inference on top of CAD/CAM). This is indicative of where we are going in the future. Think of it this way. Suppose you design a jet engine using a CAD/CAM system. Suppose that you want to modify the design to incorporate a high-bypass turbofan. This would change the gear ratios in the rear. Currently, the designer has to make such changes manually. There is no reason that the computer cannot make such changes for him or her, which frees the ME to think creatively and explore far more possibilities than would otherwise be practical. Such technological improvements are based on the capture, insertion, and utilization of knowledge. This technology is not inexpensive, but once a shell is written, it can be widely disseminated for a high ROI. A key principle derives from randomization. That is, the human should never have to do anything, or anything resembling anything similar, twice. We supply novel creative knowledge and the machine replays the ever higher-level routine knowledge. The promise of AI is to free MEs from non-creative work, while increasing their productivity and the quality of their work concomitantly.

Will we ever design objects by picking up a microphone and speaking our wishes? No, I doubt this because natural language does not facilitate the communication of formal constraints. But, at the same time, we will be able to say click and drag to enlarge the front turbofan and see a myriad of changes automatically propagate through the design as mediated by acquired design rules. One will be able to more or less formally state, optimize the size of the front turbofan to (1) minimize fuel consumption, (2) minimize noise, and (3) maximize thrust – in that (or different) order(s). Or, the required constraints on 1-3 can be input and the system given the command to optimize the design such that these constraints are all satisfied. Clearly, the need for design rules is ubiquitous. But, how are such design rules acquired?

***Corresponding author:** Tebibel TB, Ecole nationale Supérieure d'Informatique, BP 68M Oued-Smar, 16309, Alger, Algeria, Tel: (619) 553-3554; E-mail: t_tebibel@esi.dz

Received December 18, 2015; **Accepted** January 20, 2016; **Published** January 22, 2016

Citation: Rubin SH, Tebibel TB (2016) The Future of Intelligent Design. J Appl Mech Eng 5: 190. doi:10.4172/2168-9873.1000190

This is a salient question because knowledge acquisition is the most costly problem needing solution in design automation. It is our belief that such systems will acquire knowledge by doing. That is, as these systems are used to design complex devices, they will acquire cases and their generalizations for replay. Indeed, we have Phd students working on the generalization problem at the present time. It is clear that generalization requires knowledge, but at least two things are not so clear. First and foremost, it is not clear how to best acquire that knowledge and second, it is not clear how to best represent that knowledge. That is, should the knowledge be compiled by experts, which is costly – especially when one factors in the need to maintain and update that knowledge. Otherwise, as we advocate, the knowledge can be acquired by doing. The difficulty here is that validity or optimality is not assured – particularly if the knowledge is to be shared/distributed. In some areas of ME, reliance upon such acquired knowledge can potentially be catastrophic (e.g., the design of safety devices). However, this can minimize costs – including that associated with maintenance and update. A solution here is to certify that knowledge prior to distribution. This fuses the best of both worlds.

Second, the question of how to best represent knowledge for ME design is summarized by, "no single representation can suffice for all uses". We prefer cases because the many nuances of ME design are too varied to be readily captured by rules written by humans. Rather, we are looking into using rules to generalize cases. These rules are themselves subject to modification for use in generalizing other case domains. Many other types of AI – including neural networks, genetic algorithms, and the like are inappropriate because their acquisition is inherently *NP-hard*, or intractable. Furthermore, we believe that the opportunity for reuse among cases, including their instantiation, allows for the design of intelligent systems, which can propagate design transformations by case substitution. For example, if one were to substitute a thermopile for a Carnot cycle compressor, the thermostat could be substituted for by one that allows less fluctuation in temperature. This is because the latter thermostat will draw excessive and unwarranted energy if associated with a Carnot cycle compressor (i.e., due to induction upon starting). Such substitutive transformations are based upon accumulated expert knowledge.

In summary, we overviewed the past and the current traditional methods for the engineering of complex mechanical designs. We explained what the human does best – novel creative engineering and what the machine does best – repetitive, symmetric reuse. The symbiotic joining of human and machine promises to revolutionize the practice of ME by freeing the design engineer to work non-procedurally and enabling him/her to exercise creative judgement, unencumbered by today's need for repetition in the work environment. The complexity of ME designs will continue to increase along with their novelty concomitant with use and the overall growth in computing power. In actuality, the coming revolution in ME will parallel the coming revolution in software engineering upon which it depends.