Characteristics of Computing and Informatics

JÜRGEN F. H. WINKLER
Friedrich-Schiller Univ., Inst. of Computer Science,
D-07740 Jena, Germany
http://www1.informatik.uni-jena.de

1. Introduction

After fifty years the new field of computing is still a rather new discipline in the world of engineering and science [DM 97]. It has evolved quite rapidly and it seems not to have found its proper place in the world of science and engineering. It even seems not to have found its proper name. Traditionally, it is called „computer science“ which does not completely reflect its current contents. Other names are „computer engineering“ or „computing“ [CDG 89], „information processing“ (e.g. in IFIP) or „informatics / informatique“ (Denning e.g. mentions „bioinformatics“ [Den 97: 274]). A good name would also be „computational engineering“ in analogy to „electrical engineering“ or „mechanical engineering“. Sometimes it seems not completely clear what kind of field computing is [CDG 89: 9; Ber 96].

Characteristics of Computing

Finiteness
SW: Discreteness
Effectivity / Efficiency
High complexity of finite constructs
Taking Error Situations into Account

SW: Design = Product
SW: No leakage and spurious effects

Similarities to the Human Brain

In this paper we are not concerned with the name for the (new) discipline of computing but with its basic characteristics and distinctions, i.e. in this paper we try to identify the basic characteristics of this new field and its relations to its neighbouring disciplines.

There are three groups of characteristics and distinctions: the first deals with relations between computing and mathematics, the second deals with relations between computing and other fields of engineering, and the last is a general characteristic of computing.

Some characteristics are labeled with „SW“ because they seem to apply especially to programs or software. They may also apply to HW but in this paper they are discussed with respect to SW. This is no strong restriction because „most of the distinctions of computing are embodied in programming notations.“ [CDG 89: 11] and most computational engineers work in SW [GI 92]. The computer itself, i.e. the hardware, is typically not built by computational engineers but mostly by electrical and mechanical engineers.

2. Characteristics of Computing and Informatics

Finiteness: All computers are finite machines. In that computer science is different from mathematics, which has been dealing mainly with infinite constructs, especially for the last 130 years ([Mes 67; Rus 84: 783, 784], „Mathematics is the science of the infinite,...“ says Hermann Weyl [DH 81: 108]). On the other hand, Donald Knuth published a paper with the title: „Mathematics and Computer Science: Coping with Finiteness.“ [Knu 76]. In a brochure of the department of computer science of the Technical University of Munich we read: „Genuine transfinite theories are irrelevant to computer science, only the (nonfinite) border of the realm of finite theories is directly useful.“* [TUM 92: 21; transl. by JW].

Discreteness: From the viewpoint of software the level of bits is the level of greatest detail which is taken into account. A bit is conceived as indivisible, and this led Heinz Zemanek, the leading Austrian computer pioneer, to the remark „Apart from the bit there is no atom.“ [Zem 92: 274]. The world of the computational engineer is therefore a discrete one. In this world the continuum of the real numbers does not exist. Whether it exists at all is for a computational engineer a more philosophical question; modern physics also sees the world rather as a discrete one [Mes 67: 218].

Effectivity / Efficiency: Like engineers of other branches of engineering computational engineers develop things to be used by other people. In case of the computational engineers the products are mostly programs (SW). The result of the activity of the computational engineer has therefore to be a

thing which can be used *effectively*; a pure existence proof (as in mathematics) is not sufficient. Additionally, *efficiency* is often of great importance. If a program needs 36 hours to compute a 24 hour weather forecast then this program is not well suited for use in the daily weather report. Such aspects have also been mentioned by Heinz Zemanek [Zem 91: 178].

High Complexity of Finite Constructs: Programs belong to the most complex objects men have created. The complexity of a construct depends on two things: the number of components it consists of and the number of relations and interactions between those components. With respect to the number of components the constructs of computational engineering are similar to those of other branches of engineering. A program may consist of several million instructions and a microprocessor of several million transistors (e.g. Pentium or Pentium Pro). The Great Wall in China (ca. 6,400 km long) also contains a large number of bricks, even taking into account that it does not entirely consist of masonry. The main difference between the constructs of computational engineering and other branches of engineering is in the relations and possible interactions between components. In a mechanical system only adjacent components may interact. In an electrical system also components, which are not adjacent, may interact e.g. by means of a wire. But since wires need some room the number of such interactions is limited. On the other hand, in a program consisting of 10 million lines of code line 45,678 may interact with line 7,123,456. Often such interactions in a program are not explicitly expressed but only implicitly, if e.g. both lines refer to the same variable; i.e. these interactions are barely visible which makes the understanding of such constructs even more difficult: „If you try to inspect all but the smallest program, the complexity overwhelms you." [Par 94: 272]. Furthermore, the complexity of SW is increased by the fact that the interactions between components can be of rather different kind, whereas the (few) interactions between the components of the Chinese Wall are rather uniform: „The real measure of complexity is not the number of elements but the number of (nonuniform) interactions." [Fra 97: 52]

Taking Error Situations into Account: Whereas mathematicians usually can think up their constructs in an undisturbed world of pure thinking in such a way that they look *elegant* and *beautiful*[†], the computational engineers have also to take aspects of the real world into account. One prominent example for this is the reaction to errors, e.g. in the form of erroneous operation. In mathematics division for numbers is defined in such a way that division by zero is not allowed. On

---

[*] „Echt transfinite Theorien sind für die Informatik belanglos, lediglich der (nicht-finitäre) Rand des Reichs finitärer Theorien ist direkt nützlich."

[†] „If a mathematical theory has not yet proved useful, then it should at least be beautiful." [Pie 87: 311]

the other hand, a pocket calculator must be designed in such a way that such input is also processed. It is not possible to exclude such an erroneous input „by definition"; it may just be the result of some inattention of the user. It may well be the case that such error handling accounts for 50 % of a program.

SW: Design = Product: Programs are the result of the design activity of (SW) engineers. In comparison to other engineering designs programs have the following special property: they can be immediately brought into use. I.e. the realization threshold (= production cost) is essentially zero. This is in sharp contrast e.g. to the design of an architect which - in order to be brought into use - has to be realized at the construction site. The cost of construction is usually much higher than that of the design. The situation is similar for the designs of mechanical engineers (e.g. a car) or electrical engineers (e.g. a power plant). Even if the design of the SW engineer is a mass product the cost of replication is rather low[‡]. As a consequence of the low realization threshold of programs modifications, corrections or improvements can often be carried out quite easily. This is actually the reason for calling programs „soft"ware. A high realization threshold, on the other hand, implies that before implementation or realization a design is first checked and scrutinized with respect to possible errors or design flaws. Because of the very low realization threshold for programs such checks and inspections are often not done (edit-compile-run cycle). As a consequence SW products quite often contain errors [Gib 94].

SW: No leakage and spurious effects: As already mentioned above, the world of SW is a discrete one: a bit is either 0 or 1; there are no leakages or spurious effects [CM 80: 11, 12], fluctuation of properties of material, or effects of wear and tear[§]. In this aspect the life of the SW engineers is easier than that of their colleagues in neighboring disciplines.

Similarity to the human brain: Of all the artifacts made by men computing systems, especially those from the field of artificial intelligence, approximate the human brain and its accomplishments most closely. This seems to be a unique characteristic of computing. Just mentioning the term „artificial intelligence" may sometimes lead to heated debates. The main reason for this is that intelligence is very often seen as that aspect of man which distinguishes man from the rest of nature.

[‡] „Although duplication of software might be considered to be manufacturing, it is usually a relatively trivial process." [Lev 95: 22]

[§] „Since software is pure design, there is no need to worry about the random wearout failures found in physical devices ...." [Lev 95: 27]

This aspect of computing seems to lead to endless debates in which neither side is able to convince the other [see e.g. Dre 79, Sci 90]. There is not enough room to cover these discussions. Therefore I mention just three highlights from the history of computing with respect to the question at hand.

Konrad Zuse, one of the early computer pioneers, wrote already in 1937 in his diary: „Crucial idea 19. June 1937 - discovery that there are elementary operations, from which all operations of computation and of thinking can be built up. ... to solve all tasks of thinking which may be covered by mechanisms." [Zus 93: 41[**]; transl. by JW]. One specific example in this broad debate on artificial intelligence is the development of chess programs. Again, Zuse very early recognized the capabilities of the computer when he wrote in 1938: „In fifty years the world chess champion will be defeated by a computer." [Zus 93: 45]. It did not happen in 1988 but in 1997: „Garry Kasparov's legendary resolve broke down Sunday in a defeat by the IBM supercomputer Deep Blue that created chess history -- the first time a program has triumphed over a reigning world champion in a classical chess match." [Reu 97a]

3. Conclusion

The characteristics and distinctions presented in this paper mostly characterize computing and informatics in relation to its neighbours. It would be useful to complement them by some characteristics describing more directly what computational engineers are doing. One problem with this is that there exists not yet a good definition of „information". Heinz Zemanek mentions 15 definitions of „information" [Zem 92: 10 .. 18] and 7 of „computer" [Zem 92: 4 .. 8].

4.References

Ber 96 Berman, Michael A.: OO Techniques in the Classroom. ACM SIGPLAN Notices 31, 2 (1996) 4 .. 5

BR 94 Brunnstein, K.; Raubold, E. (Hrsg.): 13th World Computer Congress 94, Volume 2. Elsevier Science B.V. 1994.

CDG 89 Denning, Peter J. (Chairman); Comer, Douglas E.; Gries, David; Mulder, Michael C.; Tucker, Allen; Turner, Joe E.; Young, Paul A.: Computing as a Discipline. CACM 22,1 (1989) 9..23

Den 97 Denning, Peter J.: How We Will Learn = [DM 97: 267 .. 286]

DH 81 Davis, Philip J.; Hersh, Reuben: The Mathematical Experience. Birkhäuser, Boston etc., 1981. 3-7643-3018-X

DM 97 Denning, Peter J.; Metcalfe, Robert M.: Beyond Calculation - The Next Fifty Years of Computing -. Copernicus, New York, 1997. 0-387-94932-1

[**] „Entscheidender Gedanke 19. Juni 1937 - Erkenntnis, daß es Elementaroperationen gibt, in die sich Rechen- und Denkoperationen auflösen lassen. ... sämtliche Denkaufgaben zu lösen, die von Mechanismen erfaßbar sind."

Dre 79 Dreyfus, Hubert L.: What Computers Can't Do. Harper Collins, New York, 1979. 0-06-090624-3

Fra 97 Frankston, Bob: Beyond Limits = [DM 97: 43 .. 57]

GI 92 Fachausschuß 7.4 der GI: Zur Berufssituation der Informatiker 1991. Informatik Spektrum 15,6 (1992) 335..351

Gib 94 Gibbs, W. Wayt: Software's Chronic Crisis. Scientific American 271,3 (1994) 72..81

Knu 76 Knuth, Donald E.: Mathematics and Computer Science: Coping with Finiteness. Science 194, 4271 (1976) 1235..1242

Lev 95 Leveson, Nancy G.: Safeware - System Safety and Computers -. Addison-Wesley, 1995. 0-201-11972-2

Mes 67 Meschkowski, Herbert: Probleme des Unendlichen - Werk und Leben Georg Cantors -. Friedr. Vieweg & Sohn, Braunschweig 1967

Par 94 Parnas, David Lorge: Inspection of Safety-Critical Software Using Program Function Tables = [BR 94: 270..277]

Pie 87 Pietsch, A.: Eigenvalues and s-Numbers. Akad. Verlagsgesellschaft Geest & Portig K.-G., Leipzig, 1987. 3-321-00012-1

Reu 97a Deep Blue Humbles Chess Champion Kasparov. Reuters, May 11, 1997 11:24 PM EDT; (in PointCast)

Rus 84 Russel, Bertrand: A History of Western Philosophy. Counterpoint, London, 1984. 0-04-100045-5

Sci 90 Scientific American: Artificial Intelligence: A Debate. Scientific American 262, 1 (1990) 19..31

TUM 92 Technische Universität München: Fakultät für Informatik. München 1992

Zem 91 Zemanek, Heinz: Weltmacht Computer - Weltmacht der Information. Bechtle Verlag, 1991. 3-7628-0492-3

Zem 92 Zemanek, Heinz: Das geistige Umfeld der Informationstechnik. Springer, Berlin usw. 1992. 3-540-54359-7

Zus 93 Zuse, Konrad: Der Computer - Mein Lebenswerk. Springer, Berlin usw., 3. unv. Aufl. 1993. 3-540-56292-3