Kant Yearbook (Kantovski sbornik), **19**, 131-137 (1995) - AI & Society **9** (1995) p. 396-401.

KANT AND THE SOFTWARE CRISIS[1]

Suggestions for the construction of *human-centred* software systems[2]

MARCO C. BETTONI

1.What can I know ?
2.What ought I to do ?
3.What may I hope ?
Immanual Kant, 1787.

ABSTRACT

In this article I deal with the question "How could we renew and enrich computer technology with Kant's help ?". By this I would like to invite computer scientists and engineers to initiate or intensify their cooperation with Kant experts. What I am looking for is a better "method of definition"[3] for software systems, particularly for the development of object-oriented and knowledge-based systems. After a description of the "software crisis", I deal first with the question why this crisis could not yet be overcome. A way out of this software crisis can be expected from systems which are adapted to the human faculty of thinking. I show which foundation is in my opinion necessary and sketch the principles according to which a "human-centred" method of definition for such systems could be developed on that foundation with Kant's help.

Keywords: Software Crisis, AI Software Engineering, Human-Centred Software Design, Kantian Philosophy, Constructivism, Human Thinking Models.

1. INTRODUCTION

Up to now computer science in general and Artificial Intelligence[4] in particular have been dominated by *technology-centred* approaches. But this makes it intrinsically difficult to develop systems which are adapted to the human faculty of thinking [Bettoni & Bernhard, 1994]. In my search of ideas for the construction of *human-centred* software systems I came across Kant's work[5], especially his models of knowledge and reasoning [Kant, 1787, 1790].

[1] Not the business crisis on the software market but a quality crisis.

[2] Revised version of a paper presented at the VI. Kant Workshop (Section 4: Logical Kant studies), University of Kaliningrad and Russian Kant Society, September 21-24, Svetlogorsk (Königsberg/Kaliningrad, Russia).

[3] "Definition" collects here 3 phases: specification of the requirements, of the system concept and of the preliminary design [Hubka, 1984].

[4] AI means here that discipline of *computer science* whose main purpose is to realise on the computer *cognitive* functions, processes and performances "at which, at the moment people are better" [*Rich, 1983*].

[5] Kant distinguishes in the human being 3 *cognitive* dimensions [Kant, 1787, p.833] which are used as follows in my approach: 1. the *knowledge* dimension (mental aspects, as the faculty of thinking), 2. the *ought-to-do* dimension (ethical-practical aspects, here for instance participation, work organization, etc.), 3. the *hope* dimension (results, which I may expect if I do what I ought to).

2. THE SOFTWARE CRISIS: FACTS, DEFECTS AND STEPS TAKEN

About 20 years ago when I had my first experiences in the use of large computer systems and software products at the Computer Center of the ETH Zürich, I quickly realised that the quality of those products fell far shorter of that I required. But the majority of the users seemed at that time to be happy.
To me, up to now, although hardware has become much more powerful, the situation has not really improved. For the majority of the users instead it has become worse: although their requirements have become much more demanding, the quality of software products did not increase very much.
This *software crisis*, that is the fact that the quality of current software is in general[6] insufficient, has been recently reported by A. Sage and J. Palmer [Sage & Palmer, 1990, p.20]. According to these authors currently only 2% of the software produced can be used productively after delivery, 3% can be used only after modifications, 19% requires major redesings, 47% is no longer used a short time after delivery - 21% because it is unsound, 26% because it is incorrect - and 29% is so bad, that it is undeliverable.
Many other studies on software productivity, which analyse this situation, mention the following central defects:

> 1) software is expensive, 2) it very often greatly exceeds planned costs, 3) its delivery is too late, 4) its documentation is poor, 5) it cannot be integrated, 6) it cannot be moved to a new environment or 7) be improved to evolve along with the user's needs, 8) its maintenance is difficult and error-prone, 9) the performances are unstable, 10) they do not correspond to specifications, 11) the software capacities fall short of what is expected and demanded, 12) software is difficult to use and 13) the system and software requirements used to develop the software system do not correspond to the needs of the users.

The same studies connect these defects with steps taken in the software production, i.e.: A) with methods and tools of development, B) with methodology, C) with project management. The discipline which collects all these 3 domains and has the task of showing ways and means for the improvement of product quality, is called *Software Engineering*. But as the current software crisis shows, software engineering has not yet been successful at overcoming the problems noted above. In my opinion one major reason lies in an incomplete analysis of the situation, in which one explores only defects and the corresponding steps taken, whereas the approaches which lie behind these steps are almost never considered: but it is precisely the analysis of these approaches which cannot be avoided, if one wants to improve the steps which depend on them.

3. CAUSES OF THE SOFTWARE CRISIS

Many of the software defects just cited have to do with approaches and steps taken in the domain of *methods of definition*. In this area great efforts have been put forth as it is shown by the very widespread methods of object-oriented and knowledge-based software development. These methods should contribute to improve software quality by making possible systems which are adapted to the human faculty of thinking.

[6] Client specific, domain specific and standard software systems.

Their main strength lies in the fact that *modelling* plays a primary role in them [Coad & Yourdon, 1990, p.9]. But the approach to modelling as well as the underlying approach to knowledge has not yet been questioned critically enough in these methods. This is why, up to now, it has not yet been possible to exploit the main strength of these two methods: this is in my opinion the main methodological reason why many essential software defects still exist, although these methods are very widespread.
Two examples may show you the currently established approach to modelling in software engineering:

1. In his report to the project "Information model for manufacturing process control" - which has been developed with a mixed method, both object-oriented and knowledge-based - the author writes [Dangelmeier, 1993, p.222]: "*If one 'lifts the roof off the factory' - so to say -, then one will find objects ready-made, with different values for their characteristics, that is with different states.*"

2. The author of the book "Object-orientied software development", a world famous software expert, takes a position concerning the controversial question, how objects of a system can be found. He writes [Meyer, 1990, p.55]: "*In the physical and abstract reality objects are modelled and waiting to be read.*"

For these two authors, modelling is merely the "finding & picking up" of objects that are waiting for us ! But, if objects are merely waiting to be found and picked up, then this necessarily implies the hypothesis that the *order of things* which is embodied in our knowledge exists independently of us[7].
This position, which I would call, with reference to Kant, *cognitive dogmatism* [Kant, 1787, p.XXXV], entails or at least grounds the conviction that a good model must be an exact picture of that order of things. If this approach to modelling is used in software engineering, that is in the relationships between developer, user and software system, then this yields the following implicit requirements:

1. the user must explain to the developer the "order of things";
2. the developer must reproduce the "order of things" in the system;
3. the system must reflect the "order of things" to the user.

A critical review of the practice shows on the contrary that with current object-oriented and knowledge-based methods we do not succeed in fulfilling these 3 requirements. Each of the 3 relationships presents us with a huge gap:

1. a gap between user and developer: the user has great difficulties in presenting his requirements and expert knowledge in such a way that the developer can understand them.
2. a gap between developer and software system: the developer has great difficulties in putting into a system model the expert knowledge and the requirements of the user.
3. a gap between software system and user: the system is not well enough adapted to the user's faculty and ways of thinking.

[7] This hypothesis constitutes - as far as I can see - the foundation of the established sciences (knowledge production) and technologies (knowledge application).

This shows that object-oriented and knowledge-based methods cannnot yet fulfill their aim, which consists in making systems possible which are adapted to the human faculty of thinking, and it leads me to presume that the main methodological reason for this lies in *finding an picking up* as a modelling approach.

4. A *HUMAN-Centred* MODELLING APPROACH

If *human-centred* software systems have to be developed, i.e. systems which are as much as possible adapted to our faculty of thinking, then a new modelling approach must be devised. If this does not succeed, then the developer will hold on to the traditional attitude "I am constructing a machine" [Wirth, 1984, p.48], and this will in turn strengthen the domination of *technology-centred, machine-based* concepts.
The foundation of my modelling approach was formulated already more than 200 years ago by Kant in his "Critique of pure reason" as follows:

> *"Hitherto it has been assumed that all our knowledge must conform to objects. But all the attempts to establish something a priori about objects by means of concepts, ..., have, on this assumption, ended in failure. Let us therefore make trial whether we may not have more success ... if we suppose that objects must conform to our faculty of knowing."* [Kant, 1787, p.XVI].

For my purposes Kant's suggestion to suppose that objects must conform to our faculty of knowing constitutes the *gateway* to his Critique. The basic knowledge principle I derive from it is as follows:

> "The order of things an systems which is embodied in our knowledge depends on us and is made by us: it is aligned with (conforms to) our way of processing knowledge (our faculty of thinking)."

This position, which approaches Ernst von Glasersfeld's "Radical Constructivism" [von Glasersfeld, 1987], is also the basis of my modelling principle:

> "A good model is not the picture of an independent order, but a *working*[8] *(viable) formalisation* of the order which we ourselves generate in knowledge."

By means of these 2 principles I can now reformulate the previous examples as follows:

1. "If we 'lift the roof off the factory', then we will not find any ready-made order, but we will only *obtain that order* - i.e. systems, objects, characteristics and states - which we ourselves generate in knowledge."

2. "The physical reality is not already modelled in terms of objects. Objects are given to us as objects of experience only after we have *fixed an order* for them in knowledge: we can 'read' objects only after we have ourselves 'written' them."

[8] A formalisation which fulfills the aim for which it is beeing used.

In knowledge-based methods (systems) the model is called *knowledge base* and is composed mainly of so-called *facts* and *rules*. The established modelling approach is here the same as in object-oriented methods[9]: modelling is conceived as a *finding and picking up* of states of affairs ("Sachverhalte") which exist independently of us. My modelling approach, which I call "construct-oriented modelling", considers facts and rules of a knowledge base as working formalisations of "states of affairs" and these states of affairs as "viable constructs", i.e. as something which we make (construct) ourselves through our thinking, through the organisation of our interactions with the environment in a way which works as well as possible (viable way).

5. RENEWING AND ENRICHING SW-DEVELOPMENT WITH KANT'S HELP

The principle of construct-oriented modelling constitutes the first building block of my method of definition for software systems: in my work as a software and knowledge engineer I have already experienced many times that it does work. A second building block, which is tightly connected to the first consists in the following hypothesis:

> "The approximation, in the computer, of the way in which we generate an 'order of things' in knowledge, is a necessary adaptation of software systems to the human faculty of thinking."

A model of the way in which we - by means of our knowledge processing - make an order of things, must fulfill two basic requirements. They are:

1. Knowledge processing must be considered primarily as *synthetic-constitutive* (instead of analytic-transformative).

2. Any system (*living or non-living*), which performs primarily synthetic acts, must be modelled as an *organism*, not as a machine (organism as a condition of synthesis).

This is now the place where computer technology could be further improved with Kant's help[10]. In the "Analytic of concepts" Kant develops a theory of mental activity which - in my interpretation - fulfills the first requirement [Kant, 1787]; and in the "Critique of teleological judgement" I see that the foundations which are necessary to fulfill the second requirement have been sufficiently elaborated and could be used to design an artificial, *non-living* organism [Kant, 1790].

6. CONCLUSION

We still have a long way to go in order to develop with Kant's help a *human-centred method of definition* for software systems. My concern in this article has been primarily to show the direction we must travel, in the hope that this contribution may stimulate computer scientists, engineers and Kant experts to initiate an exicitng transdisciplinary cooperation.

[9] "The *real* world is made of objects and their relations. The corrspondent *symbolic* world is made of object symbols and of relational symbols." [Lusti, 1990, p.17].

[10] Currently AI cannot yet fulfill these 2 requirements because its approaches are either restricted to the *transformative paradigm* (analytical-transformative in symbolic AI and emergent-transformative in Connectionism) or strictly consider that synthetic-constitutive processing can take place only in *living* organisms [Bourgine & Varela, 1992].

REFERENCES

Bettoni, M. & Bernhard, W. 'Simulation with MASTER'. In: Liebowitz, J. (ed.), Moving Towards Expert Systems Globally in the 21st Century, Proc. 2nd World Congress on Expert Systems, Lisboa (P), 10.-14. Jan. 1994, Macmillan New Media, Cambridge (MA), 1994.

Bourgine P. & Varela, F. (eds.), Proc. 1st European Conference on Artificial Life (ECAL '91), MIT Press, Cambridge, MA, 1992.

Coad, P. & Yourdon, E., Object-Oriented Analysis. Prentice Hall, Englewood Cliffs, NJ, 1990.

Dangelmeier, W., 'Objektorientierter Modellierungsansatz für eine regelbasierte Fertigungssteuerng.', Zeitschrift für wirtschaftliche Fertigung und Automatisierung ZwF, 88 (1993) 5, 222-225.

Hubka, V., Theorie Technischer Systeme: Grundlagen einer wissenschaftlichen Konstruktionslehre. 2. Aufl., Springer-Verlag, Berlin, 1984.

Kant, I., Kritik der reinen Vernunft, Riga, 1787. I.Heidemann (Hrsg.), Reclam, Stuttgart, 1966.

Kant, I., Kritik der Urteilskraft, 1790. W.Weischedel (Hrsg.), Werkausgabe, Band X, Suhrkamp, 5.Aufl., Frankfurt a/M, 1981.

Lusti, M., Wissensbasierte Systeme: Algorithmen, Datenstrukturen und Werkzeuge. BI, Mannheim, 1990.

Meyer, B., Objektorientierte Softwareentwicklung, Hanser Verlag, München, 1990.

Rich, E., Artificial Intelligence. McGraw-Hill, New York, 1983.

Sage, Andrew P. & Palmer, James D., Software Systems Engineering. John Wiley & Sons, New York, 1990.

von Glasersfeld, E., The construction of knowledge. Intersystems Publications, Salinas, CA, 1988.

Wirth, N., 'Data structures and algorithms'. Scientific Am., 251 (1984) 3, 48-57.