Thinking through computing

David Clark
UCL Centre for Publishing, SLAIS
University College London
www.publishing.ucl.ac.uk
d.j.clark@ucl.ac.uk

this is a workshop...

The programmer, like the poet, works only slightly removed from pure thought-stuff. He builds his castles in the air, from air, creating by exertion of the imagination. Few media of creation are so flexible, so easy to polish and rework, so readily capable of realizing grand conceptual structures ...

Yet the program construct, unlike the poet's words, is real in the sense that it moves and works, producing visible outputs separate from the construct itself. It prints results, draws pictures, produces sounds, moves arms. The magic of myth and legend has come true in our time. One types the correct incantation on a keyboard, and a display screen comes to life, showing things that never were nor could be.

Frederic P. Brooks, Jr. *The Mythical Man-Month. 1975*

Analogue, Analog, Digital,

The development of solid state devices and ferrite core memories reinforced the conception of the computer as a discrete and logical device for the solution of digitised problems. The development of programming languages engendered a new form of Cartesian dualism: the division of hardware and software.

Metaphor or Epicurian *Prolepsis*

If I had merely to describe to you some new discovery in science, I should be able to avail myself of your previous knowledge as a foundation, and to erect thereon a representation of the new fact which you were to place beside those old ones which you knew before. The greater your previous knowledge, the easier would be my task. But what I have to do is something quite different. I have to shew you the facts with which you are already acquainted in a light of a different character from that which the most illustrious philosophers have shed upon them—a light which the wisest among them would probably have avoided as deceptive and misleading had it been presented to him in his own time, because the slow yet steady progress of science has only in more recent times prepared us for its reception.

on Faraday's Lines of Force, James Clerk Maxwell 1873

Mathematicians or Engineers

The dominant academic status of mathematics, well established in the mediaeval universities and reinforced by Renaissance neo-Platonism. By the nineteenth-century a firm foundation in mathematics was seen as the mark of a mature science, thus physics became the exemplar of a ‘complete’ science and throughout the twentieth century motivated a remaking of biology in the manner of physics. The establishment of a ‘profession’ of engineering has likewise demanded the promotion of mathematical over empirical methods.

Biology or Physics

- Biology: automation as biota: objects embedded in environment, exhibiting behaviour. How it works; a potentially tractable simplified model of real life.
- Mathematical-Physical: computers and automata as agents of ratiocination, representing the world symbolically. The process of problem solving detached from sensation and action. Transition across this Cartesian divide—problems of deixis and reference, and a distinction between understanding and knowing—were largely ignored or reduced to a subsidiary instance of problem solving.

Principles or Practice

In their capacity as tool, computers will be but a ripple on the surface of our culture. In their capacity as intellectual challenge, they are without precedent in the cultural history of mankind.

EW Dijkstra. '*The Humble Programmer*' ACM Turing Lecture 1972

Builders or Gardeners

Business people are comfortable with the metaphor of building construction: it is more scientific than gardening, it's repeatable, there's a rigid reporting hierarchy for management, and so on. But we're not building skyscrapers—we aren't as constrained by the boundaries of physics and the real world.

The gardening metaphor is much closer to the realities of software development. Perhaps a certain routine has grown too large, or is trying to accomplish too much—it needs to be split into two. Things that don't work out as planned need to be weeded or pruned.

Andrew Hunt, David Thomas.
The Pragmatic Programmer: from journeyman to master. 2000

Programmer to User = Author to Reader

An interaction between users and designers of such a virtual machine occurs at a virtual machine level. There is, effectively, a communication between programmer and user mediated by virtual machines. Moreover, this relation may be characterised as a literary one—between author and reader, between a text and its context.

Program as literature 1

To discuss the literary qualities of a text is to consider how a particular kind of ‘virtual machine’ is made to do its work. It is in this context that the program may be appreciated as a literary artefact, the programmer as author and impresario, the computer an actor performing a part that is written. What matters is not the book in hand, not the staging, but what goes on between reader and writer, an interpretation of a shared text.

Program as literature 2

Programming is, like any form of writing, more often than not experimental. One programs, just as one writes, not because one understands, but in order to come to understand. Programming is an act of design. To write a program is to legislate the laws for a world one has first to create in imagination. Only very rarely does any designer, be he an architect, a novelist, or whatever, have so coherent a picture of the world emergent in his imagination that he can compose its laws without criticism from that world itself.

[...] The relationship between understanding and writing thus remains as problematical for computer programming as it has always been for writing in any other form

Joseph Weizenbaum. *Computer Power & Human Reason*. 1976

Program as literature 3

When software became merchandise, the opportunity vanished of teaching software development as craft and artistry. The literature becomes frozen. It's extremely rare today to stumble across someone who is familiar with the same source code as you are. If all remnants of literature disappeared, you'd expect that eventually all respect for it—as an art form, as a craft, as an activity worthy of human attention—would disappear. And so we've seen with software: the focus is on architecture, specifications, design documents, and graphical design languages. Code as code is looked down on : the lowest rank in the software development chain is the 'coder'—right alongside QA drone and doc writer

Richard P. Gabriel, Ron Goldman. 'Mob Software: The Erotic Life of Code' ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications 2000

Virtual Machine or Nominalism Realised

The computer is unique among tools in its nature as a virtual machine that crosses the established boundary between mechanism and mind, and thus well-established social and intellectual classes of mental and physical labour.

Pure thought stuff?

- education
- training
- bildung
- wissenschaft
- philosophy
- scientia

- craft
- art
- trade
- profession
- vocation
- métier

Logical Positivism's last redoubt

Counting is the religion of this generation it is its hope and its solution.

...That is the reason that everybody thinks machines are so wonderful they are only wonderful because they are the only thing that says the same thing to any and every one and therefore one can do without them, why not, after all you cannot exist without living and living is something nobody is able to understand while you can exist without machines it has been done but machines cannot exist without you that makes machines seem to do what they do. Well anyway...

Gertrude Stein *Everybody's Autobiography* 1938

that worldview for which a computing machine is a desideratum

To try to deal with all matters by logico-scientific language is as self defeating as to try to capture water in a net, or breeze in a bag.

Phillip Wheelwright *Metaphor and Reality* 1962

enclosure and completion

• three criteria that mark the formation of a distinct discipline:

•the founding of institutes and learned societies,

• debates about curricula and education,

•the formation of a distinct cognitive content.

Joseph A Caron. 'Biology' in the Life Sciences: a historiographical contribution'. *History of Science* xxvi (1988)

distinct cognitive content

Computing appears to be a science that lacks a distinct cognitive context. How, if it is the study of algorithms, is it to be set aside from mathematics? If engineering—and there is much to be said for this view—what material, what physical restraint, does it work with. Can it be neatly distinguished from, say, electronic engineering? If it is design in the abstract, can it still claim to be a science, or should it be thought an art or craft

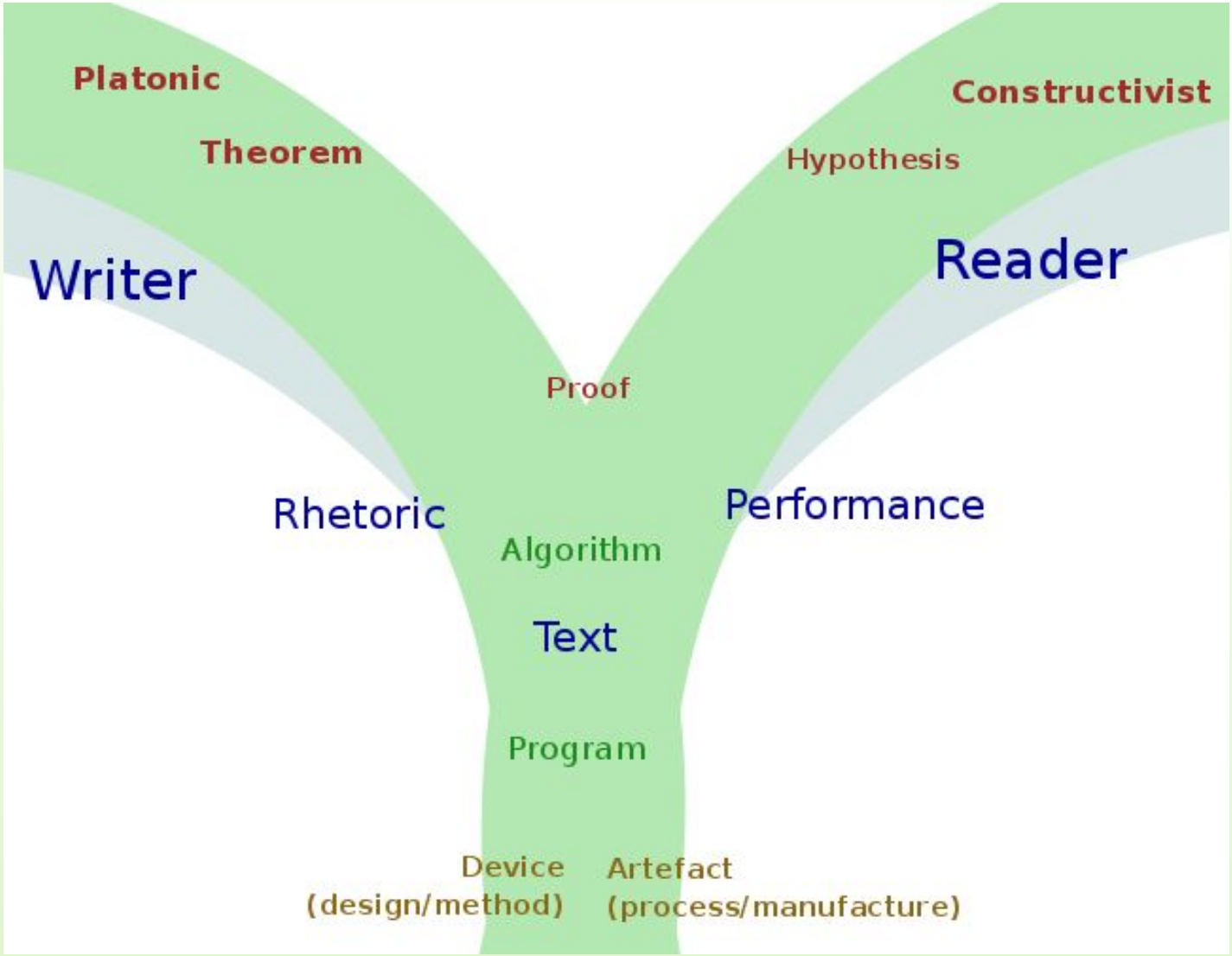
Algorithm or Program

- the algorist's virtual machine—an idealisation, on which to base a proof
- the software engineer's virtual machine—a simplification, by which a design may be rendered with greater clarity.

Writers and Makers

although the programmers activity ends when he has constructed a correct program, the process taking place under control of his program is the true subject matter of his activity, for it is the process that has to accomplish the desired effect; it is this process that in its dynamic behaviour has to satisfy the desired specification. Yet, once the program has been made, the “making” of the corresponding process is delegated to the machine.

Edsger W Dijkstra.
‘Goto Statement Considered Harmful’ *Communications of the ACM* 11(3) 1968


Platonic
Theorem
Constructivist
Hypothesis
Writer
Reader
Proof
Rhetoric
Performance
Algorithm
Text
Program
Device
(design/method)
Artefact
(process/manufacture)

What I suggest here is that programs evoke and describe virtual machines and their processes in a manner analogous to the way literary language evokes and describes ‘worlds as real as but different to the one that exists’ in our ordinary experience of life. The skill to write a program is not the same as that required to write a novel (though I think it may be of the same order); it would undoubtedly take a rare person to do both equally well. But writing programs *is creating* a virtual machine, it is ‘pure thought stuff’, it us not a science of pure abstraction in the manner of mathematics, nor does controlling machinery accurately capture what it is like to create a significant program. The program is a virtual machine, distinct from the physical computer required to instantiate it, yet that need to instantiate is significant, it is what makes a program different from an algorithm, a purely abstract mathematical form. Like successful storytellers, whose creations may suspend disbelief but cannot disregard it entirely, programmers are not free to dream anything, their creations have to work on a real (and imperfect) machine.

The first of computing's intellectual challenges then, is to recognise its own boundaries, to become, not perhaps self-conscious, but to have an independent identity. A self-image has to grow beyond emulation. The story of computing so far has been one of following prevailing social and intellectual models, most notably those of mathematics and engineering. But, as we have shown, there were, and are, other patterns and moulds. If computing has a significant part to play in cultural history, it must surely become more human centred, that is humanist; weaving into its rational heritance, the biological and the imaginative capacities of the creatures that created it.

It is in pursuit of a better model of what programming is that I would commend a study of rhetoric, though the term ‘rhetoric’ is itself problematical. Firstly, because of all the modern associations of rhetoric: with vain posturing and insincere speech. There is also the presentation of classical and renaissance rhetoric as it has come down to us filtered through the Victorian schoolroom. A nit-picking classification of ‘figures of speech’ and difficult Greek and Latin terms for what is, in ordinary fluent speech, intuitive, natural, and thereby unconscious. Secondly, we have the rhetoric of modern theory. Never entirely free from a pejorative tone, this tends analyse every thought and action as well as speech in search of an alternative reading. Rhetoric in this sense means every historical ‘actor’ is assumed to be saying one thing in order to bring about something contrary or suppressed. Nothing is to be taken as it seems, we must always suspect ulterior motives. In such a reading of history everyone begins to appear as either a helpless puppet or duplicitous schemer. None of these versions of rhetoric will do; for a model of programming as a *logon techne* it is necessary to return to the historical root and also to cast off the negative impression promoted by Socrates and Plato.

.

In this model *rhetorike* (a term probably invented by Plato and the fount of its subsequent negative image) was a response to a profound change in the relation of written and spoken word that occurred in the fifth century BC. A fundamentally oral culture became literate. The move toward literacy had begun some four centuries before but its cultural impact had been limited. The predominance of the spoken word had persisted and with it the authority of those who could speak well in public. It is probably the importance attached to speech that ensured that in adapting Phoenician script the Greeks added a unique feature: the vowel signs by which not just a sign of a word but the spoken word could be recorded. The result was a language of poetry rather than commerce. In the fifth century changes in political organisation placed ever greater importance on public speaking in the law courts and assemblies of Athens. But literacy, by enabling speech to be set down, had made it possible for public speaking to become a skill that could be taught. For the first time words could be abstracted from the stream of consciousness. Thinking became visible, analytic. The form of words that might best persuade an assembly could be studied, practised, and reproduced. A new technology, literacy, had made possible a new way of thinking and a new form of literature. Interdependent, speech and its writing had changed each other

How should this apply as a model of, or for, programming? We can see programming as an equally profound intellectual challenge: of fixing in script the flow of process and algorithm. What is fixed is a text that can be studied and imitated but there is also something intangible, a power of persuasion over machinery, a power to invoke “things that never were nor could not be.” The economic and social value of ‘computing’ has created a demand for handbooks or *techne* of our new form of words. This demand can be exploited by charlatans, misunderstood by those who feel threatened by it. But most of all it is a means, I hope, to come to grasp the essentially intangible yet real nature of a program: that is of its having *effect* in the physical and *affect* in the mental world. Programs *work*, if they do not they are mere plans or at best unproven algorithms. But they are nonetheless, in essence, mental phenomena—not shrink wrapped products, nor ‘virtual’ machinery, nor anyone’s ‘intellectual property’. It creates a world, as real as, but different, from the one that is. It is this intangible ‘work of fiction’ character of the program that is most in need of elucidation, *writing* software is like any other kind of writing