What is Software?
[areflagan@artpanorama.com]
What is software? The mechanistic answer would be that it is assemblages of algorithms compiled to perform and automate specific tasks on a computer. If we resided on the circuit board, somewhere among its rigid corridors of conduction, such a reductive definition might surmise to understand software, but only within the dark limits of the black box. Let's look outside this box for a moment. We are looking at a machine that synthesises and automates modalities of social, cultural and economic relations, removes most tinkering from its root, and installs a generalised operator to perpetuate a programme of utility valued and developed by its owners. Now that the initial question has partly been rephrased, it should be asked again. What is software? Ø

redistribution.pl
[harwood@scotoma.org]
# Perl routines for the redistribution of the world's wealth
# Takes the cash from the rich and turns it into clean drinking water wells
# w.blake@scotoma.org
# v 0.0.1 / Ø

> [areflagan@artpanorama.com]

Profiling Software
[areflagan@artpanorama.com]
/ Profiling works in the background to facilitate decisions on entry and exit when packets travel port to port across borders of difference. And profiles very explicitly invite many more such analogies, laboured or not, precisely because they are instruments of conversion that always return to the currency they carry. This leads to the thought that persistently haunts me: that profiles explain why the world makes some sense at all, how evil divides itself from good, black from white, religions from each other, one culture from the next and so on.
They are the troublesome and comforting ideological supplements that I carry around as inevitable baggage in return for knowledge and perception, those cerebral turns towards understanding and recognition. Profiles hence allow my views to converse across divides by putting the general before the specific, ideology before the individual. Through them the calibrated and manufactured impressions that encourage me to see a criminal in every black man and a terrorist in every Arab take root. Profiles are my hidden supplements to a point of view, and I honestly swear by them. Software profiles, that is. Ø

> [amcgee@freeshell.org]
Who wrote this? Where is the original article located? Å

Software/Copyright Googlism
[tripta@sarai.net]
The answers are:
software is <http://www.googlism.com/?subject=software>
copyright is <http://www.googlism.com/?subject=copyright> Ø

> [areflagan@artpanorama.com]
/ It does indicate how we may conduct our existential searches these days - through algorithmic processing with finite, programmed answers, i.e. through software. How do, for example, the omnipresent contact and calendar applications, coupled with their handy accessory, the PDA, fit into how life itself organises around data and specific software clusters? Ø
> [sva2003@columbia.edu]
/ That's what most technology advertising would like us to believe. /
Where software is decidedly different is that it can be changed. That, after all, is meant to be the point of differentiating between soft and hardwares - always an arbitrary division, of course. Å

A Case for PDA Student Computing
[human@electronetwork.org]
1. A FEW BASIC ASSUMPTIONS
(http://www.electronetwork.org/articles/psc/) Before detailing what is needed, it would help to dispel aspects of present-day computing cultures, which are not needed in classroom computing for the most general utilities of computing, in its widest sense, as a learning device, as a support tool and not an education in itself. / Therefore, for some specific applications, open source software developed within local, national and international school systems may be co-developed and shared, instead.

2. PDA STUDENT COMPUTING
/ All the issues inherent in traditional computers, from price to size to portability to troubleshooting and tech support, tend to a whole different approach to student computing.
It is proposed that PDA Student Computers (PSC) are superior on almost every level, at which students are prepared to use them, in a cost-effective way, in comparison with full-fledged computers whose resources may never get used before they are outdated.

> [areflagan@artpanorama.com]
The PDA(ish) device you refer to has been developed in India, and is currently in its production phase. It is called the Simputer. I have also read about recent cases in South Africa, where some schools turned down the usual Microsoft gifts and instead installed old 486 machines running Linux. The stated goal was to teach networking and programming, and forsake the task of simply training software operators. I think this dilemma resonates throughout your proposal, and the question returns to what sort of economy education ultimately participates in.

3. PDA COMPUTER SOFTWARE
/ As mentioned, with the hardware available today, off-the-shelf technologies could be used and designed to meet the specific needs and requirements of educational computing. They could be mass manufactured on such a scale as to bring the price-point down from those of commercial models.

4. PDA COMPUTER SOFTWARE
/ PDA Student Computers need an operating system that can use both proprietary and open source software programs. / Many of the best resources, such as dictionaries and other devices, exist in the open source community, and they would be superior in price and comparable in functionality to a commercial product, and therefore this open design is essential in the socio-economics of PDA Student Computers.

5. THE PSC DIGITAL CLASSROOM
/ In all, to summarise the PSC Digital Classroom as described, it is an available, malleable, robust, affordable, open-ended, upgradeable, efficient and perfectly sufficient computer system for use in primary and secondary schools. / It has most all the functionality that makes computers worthwhile to invest in, for student education, if the goal is to supplement the learning process, and open new horizons related to personal educational computing.

> [areflagan@artpanorama.com]
/ This is a really interesting area where I hope we can benefit from the insights and experiences garnered at Sarai. / What you propose seems to be taking shape on another continent - currently going through its own personal computer evolution. And all this is taking place while Microsoft is negotiating hefty discounts, up to 90%, to become the OS and supplier of choice in these countries. We are perhaps talking some serious ‘social aspects’ of software here.
If such a platform exists, it would allow a computer that is not controlled but controllable, which starts off open - and not closed - to independent possibilities. It is almost an issue of ‘free computing’, as in ‘free speech’. By not restricting free development, it would resist enslavement to corporate models of hardware and software development, and the ideologies which place certain things at the top of the list, and others off the table or impossible, because of special interests and potential loss-of-current-profit models.

---

Memo to the Government of Kerala

Members of FSUG-Kochi have submitted a memorandum to the Government of Kerala regarding the choice of software and syllabi prescribed for the IT@School project.

Software is Religion

Software is religion. The preachers seem to understand little about those they preach to, and the ones visiting the church understand only vaguely what the preachers are all about. Software is also religion because the unbelievers’ questions are far too often answered by saying that software moves in mysterious ways.

There are many churches scattered across the land. There is the Church of Windows, which looks nice but lots of the people visiting have trouble with the fact that bits and pieces of the ceiling tend to fall down and occasionally kill someone. There is the Church of Linux, which looks solid, yet once inside one discovers that 100 different priests all claim to be the best, and besides that the place could do with a good cleaning, too.

There’s probably a church for every application or operating system ever made and it’s
believers are as fanatic in most cases as those of everyday religions. / Software is religion, because people expect miracles and the priests are unable to deliver (yet when it's time to collect money to support the church, the miracle is sold quickly with lots of extras). Software is religion, because there's the promise that if we use it, we will become better people. Software is religion, because it comes with Bibles that explain it all to us. Software is religion, because it will always remain a mystery to the masses, and there will always be questions it cannot answer. Ø

>> [douwe@oberon.nl]
Also, since the early days, the flame wars on usenet about OS preference have been called religious. Ø

>>> [areflagan@artpanorama.com]
Perhaps an excursion into anthropology, arguably the most suspect of all 'sciences', can take the fervor of feeling associated with platform preference and software further? / Maybe this is in part due to the participatory aspects: through Photoshop, or any other package with a sizeable market share, we are all part of the same denomination, obeying the same commands, performing the same rituals. To use is to belong, and the flawless (re)articulation of algorithms is the litany prescribed and absorbed as scripture. Ø

>>>> [douwe@oberon.nl]
The open source community would also be an interesting target, with their prophets, wars and Microsoft as supreme evil. By the way, is there anything wrong with the mailing list? I posted and the article bounced. Also, there has been almost no traffic recently. Æ

> I give up. The message was undeliverable. This is a fatal error.

Human-Computer Oscillation and the Need for Calories
[harwood@scotoma.org]
(http://www.scotoma.org/mouse) The foundation of this report is the human sensing system's contribution to the physical geography of the body: the orientation of the body in space, an awareness of spatial relationships and an appreciation of the specific qualities of different places and different things, both currently experienced and removed in time. The body's sensing systems offer important media through which space and time are experienced and made sense of in a computer interface. /

TASKS
Tasks are a specific chore or duty to be done. They exist on either side of our oscillation and are pre-programmed by the user's or computer's environment. / So it can be said: the user's objectification of content in the computer interface relies on the selective translation of user force through locator devices. Software (interface) requires the
recognition of BITs on the part of the user in order to allow the user to objectify content within the interface. /

THE MOUSE
/ The mouse having no absolute origin is a relative device containing two states at the top level (working and not-working) and two parameters (movement and button-states). / This objectification [mapping of mouse input] in the interface allows the user to see software structures as part of a fixed environment that is external from themselves. / Objectification within this model is a kind of meta-system transition. / In objectification the new process controls not the old ones, but the objects representing these processes. / In interface design for instance, algorithms are defined as computational processes that we expect to be executed in a certain fixed manner by the user. Having established a reliable model of one specific aspect of objectification in interface design, the report reader may like to consider the following questions. Q: What are the consequences for the appropriation of value within capitalist systems if we interfere with this objectification process within interface design? Ø

> [areflagan@artpanorama.com]
The process has also been cast, in the Marxist critique, as one of alienation. I don't quite recognise the description that allows the user to ‘see’ this structure as fixed and external: it would effectively mean a transcendence to this reality and also, by some measure, a recovery of agency. Software is arguably a process that seeks to ‘objectify’ in favour of a certain remove. One recourse may be in the symbolic, interface design, which you do not enter into. / Ø

>> [harwood@scotoma.org]
The main focus of this question was to assert, or make apparent, the road to perfect art, in which the software will be no more then the concrete sum of its parts. The art will also be devoid of all social/political/cultural/formal or economic value. The only use of computers allowed is that of a thoroughly aimless one. / Ø

Q: Having established that the selective reading of the user's input data through the mouse helps lead to objectification of content within interface, what happens if we create software that acts on all possible variables within mouse interaction? Ø

> [areflagan@artpanorama.com]
/ In other words, what happens when we integrate the movements not anticipated and logged by software into software? / Ø

>> [harwood@scotoma.org]
I'm sorry, I can find no connection between this paragraph and the report. AE
Alternative Software/The Forbidden Fruits of Computing

[areflagan@artpanorama.com]

/ The relations Turing desired in computing were at their inception directed at one missed individual, rather than a generalised and disciplined body. They also spoke of a sexual difference – subtly restated in his choice of sex to determine human intelligence in the Turing Test – that cautiously countered the machine’s universality. Once such pensive constructs enter the very definitions garnered here – those of our relations formed with the machine through hardware and our proof of its intelligent humanity in software – their universality as modes of knowledge about the life they perpetuate is called into question. When Turing took a bite of the cyanide-laced apple [his suicide], he chewed a chunk of that encounter in the Garden of Eden, which punished curiosity when it interfered with commands; he tarnished the purity of Snow White and the gravity of Newton’s eureka that found enlightenment through reason. In computing, there should always be a lingering aftertaste, a protracted mortality, that recalls the tang of sweet and bitter juices Turing must have savoured as he pondered the fruits of his work, in that final system failure. Ø

> [pankaj@sarai.net]

/ Do you have any clue what you are writing, interpreting, mis-interpreting? / Ø

>> [areflagan@artpanorama.com]

/ The prospects of understanding computing (or software/hardware) cannot belong to maths and logic alone, unless these are also recast within their own social, cultural and philosophical histories. / Ø

>>> [supreet@sarai.net]

/ Maths and logic, if you notice, tend to be very dependable. / Ø

>>> [pankaj@sarai.net]

/ I disagree. Social, cultural and philosophical histories are subjective matters and everyone has a different interpretation of them. / Maths and logic are different, because it’s all imagined and delivered, mass hypnosis in a very orthodox form called education. 1+1=2 can I question that? OK, that was a joke. / Ø

>>>> [areflagan@artpanorama.com]

on enterQuestion
global ganswer
if 1+1=2 then ganswer = FALSE
else if 1+1=11 then ganswer = TRUE
end if
updateStage
end enterQuestion Ø

>>>>> [supreet@sarai.net]

It just struck me that to implement this code, you would need to modify
the mathematics; that is the basic operators to your choice of
language: 1+1 should not evaluate to 2. Let's make an assumption
that there is this clan or a tribe, which uses an alternate form of
mathematics. This kind of alternate mathematics could be
implemented like this:
#include <iostream>
using namespace std;
class alternativeMathematics
{
    private:
        int almv;
    public:
        alternativeMathematics(int val)
        {
            almv=val;
        }
        friend int operator+( alternativeMathematics &t1,
            alternativeMathematics &t2)
        {
            return t1.val();
        }
        friend int operator-( alternativeMathematics &t1, alternativeMathematics &t2)
        {
            return t1.val()+t2.val();
        }
        int val()
        {
            return almv;
        }
};

int main()
{
    alternativeMathematics a(1),b(1),c(0);
    cout << a-b<<endl;
    cout << a+b<<endl;
}

Modify main() function to implement if-then-else/not operation. In this
program 1+1 = 1 and 1-1 evaluates to 2. Ø
> [supreet@sarai.net]
/ It would be even better if you could point to a live problem/solution
scenario where hardware/software constraints could be made glaringly
visible. / Ø
This is the crux: the conjunctures of hardware/software/logic/maths and language/culture/society effectively seek to preclude such visibility. They seamlessly integrate their programmes into a machine operation, and thereby mechanize and, in the sense of a transcendental effect, naturalise their appearance.

A neuron also performs almost the same function as a regular transistor would. It switches on/off. Human beings are not special.

Now, that's a bunch of bullshit!!, huh!!

What would be interesting to know is what's so bullshit about it, and on bullshit scale of one to ten, where it is rated?

1 2 3 4 5 6 7 8 9 10

BUT, transcendentalists have been trying this [establishing a universalised norm] ever since the Tower of Babel and Plato's Republic, without ever being successful – because the world does not work this way or resists. So we might be tempted to say that Kevin Kelly's "God is the Machine" is a bit hyped over "the mystical doctrine of universal computation", when increasingly research notes that instantiation DOES matter, especially if "there is no outside".

The End

Halfbakery.com is a forum which allows users to initiate discussions around ideas and scripts for open source movies (until they are filmed).