

# Software Transform 3 March 1992

Ian Pearson, Futurologist

This document is provided free of charge for your own personal use, but the copyright remains the property of Futurizon GmbH. No reproduction or distribution of this article in any form is allowed unless you have written consent from Futurizon.

You can read more articles on almost every aspect of the future at <http://www.futurizon.com>. Futurizon provides speakers, consultancy, and commissioned reports on all aspects of the future.

Contact details: [info@futurizon.com](mailto:info@futurizon.com) or [idpearson@gmail.com](mailto:idpearson@gmail.com)

I really wished at the time that I'd been better at both Maths and electronics. I am still sure there is some merit in this idea, though it has never gone anywhere yet. This idea gave rise to my later approach to thinking about machine consciousness, where I came up with the Pauli Switch, Heisenberg resonator and neural vortex, two of which ideas have also arisen independently elsewhere and are now in development. I guess it also led to my design of processing gel and smart bacteria. I really should develop the idea further one day...

Concept

Software Transform

Problems

Inflexible, brittle, unreliable software which is incapable of self optimisation or evolution.

BT Opportunity

To produce reliable software which is not brittle, that can be optimised for a particular environment.

To allow conversion of algorithms between the digital and analogue domains.

To design better methods of handling analogue data

Summary

If we take a mathematical function such as a step, this can be broken down into a series of sinusoids by means of a Fourier transform. If one of the constituent waveforms is modified slightly, this will have an effect on the resultant step, but the effect would generally be graceful and non catastrophic. Virtually any function can be decomposed using Fourier transforms and other transforms of various descriptions.

Since a lot of computer software consists of code which can be broken down into functions, perhaps transforms could be designed which would allow software to be decomposed into components outside the 'line domain', i.e. Software Transforms'. These constituents individually may not be obviously useful code, but together would make up the original

function. If such transforms could be made, this would perhaps give the prospect of allowing the constituents to be modified without catastrophically altering the overall function. This transformed code would be less brittle than conventional code. By varying the constituents, better functions may sometimes result (while many other changes would be detrimental), thus giving the possibility of code optimisation, adaptability and even evolution. Such decomposed code may lend itself to parallel processing machines, each of many processors executing constituents in parallel. This may even be a way of distributing work between processors which has previously been confined to single processors.

In the future, it is possible that analogue computing will make a comeback. The capability to transform between the digital and analogue domains by using software transforms would allow some of the benefits of analogue computing to be extended to current software, thus protecting investment.

If such transforms exist, an understanding of them would doubtless assist in our dealings with analogue world, so that analogue functions could be 'inverse transformed' into the digital domain.

Ian D Pearson  
Advanced Concepts Unit  
642900  
3/3/92