Digital Writing as Speech Act – Jerome Fletcher

Digital literature aka electronic literature, e-lit, digital poetics, e-poetry, etc. is still very much in its infancy. The fact that we haven’t even settled on a name for this field of theory, discourse and practices is testimony to this. The very first digital texts were produced in the 1950s by programmers like Christopher Strachey and Theo Lutz, the former creating surreal love letters, [Comment on Strachey’s image] and the latter generating stochastic poems based on Kafka, and even then, these two were more interested in questions of randomness in mathematics than they were in making a contribution to Western Literature.

A survey of the incunabula of digital literature reveals some interesting defining characteristics of this mode of writing which have a bearing on the topic of this conference. The purpose of this paper is to explore notions of editing and dialogue in relation to compositional process in digital literature. It will argue that the digital writer only ever produces a text of radical incompleteness. This is a familiar argument of course, one that can be applied more or less to any text digital or print. Writing is a call to which reading is a response. However, I’d like to make a case for digital text/literature as a paradigm for this sort of dialogic, writing-as-rewriting. Moreover, to complete a digital text, to make it perform, requires a series of dialogues between various elements of the digital apparatus (machine, codes, etc.) and often an ‘editorial intervention’ on the part of the digital reader/user. Moreover, this paper will also argue that given the performativity of digital text, it only exists as text for the duration of its performance. Beyond that it can only ever constitute a dispersed potential. In other words, where digital literature is concerned, writing will by definition be processual, dialogic and, primarily, editorial.

I maintain that digital writing is emblematic of an approach to writing that is largely at odds with the view that Barthes espouses. The problem, or glory, of digital text is its radical openness. This is largely because it is a dispersed and processual event rather than a coherent object, which to a large extent is what the print-based book is.

So, firstly what does it mean to say that this type of writing is processual? Let’s begin with a basic tenet. Digital writing requires a digital machine to compose it, to create it. This is axiomatic, in that if a computer is not involved in the composition and display of a digital text work, it’s not a work of digital literature. the opposite is not true of course. It is quite possible to use a computer to compose a piece of print-based writing , but this doesn’t make the outcome - the printed text - a piece of digital literature. It was to make this distinction that Katherine Hayles coined the term ‘digital born’ to separate out such works of digital literature, becauseThis is not a trivial concern, in that if the computer is the sine qua non of digital literature, then we need to take it seriously in any full account we give of digital writing.

What is clear then is that even before any sort of creative/imaginative/literary writing takes place in a digital environment, there will already have taken place a number of different writing processes within the machine itself. These include the writing of the operating code as well as the programme software used to create the digital text work. Some theorists such as Florian Cramer have argued that this form of writing, as it were, doesn’t count as a form of writing because it is too instrumental, nothing but the composition of impersonal instructions - that code lacks the social dimension or engagement of natural languages.

As he points out, “…computer control language is language that executes. As with magical and speculative concepts of language, the word automatically performs the operation. Yet this is not to be mixed up with what linguistics calls a 'performative' or 'perlocutionary' speech act,” because the operations of the machine have no a priori social meaning. Software engineer Ellen Ullman concurs: “[A] computer program has only one meaning: what it does. It isn’t a text for an academic to read. Its entire meaning is its function” (qtd. in Hayles 48). However, these notions of (non)performativity engage with code as a finished product, not as an ongoing writerly process. Over against this view I would cite Scott Dexter’s *‘Towards a Poetics of Code’* and Geoff Cox’s *Speaking Code.*

**Scott Dexter (coding as social practice - process and dialogue) Geoff Cox (coding as expression and politics)**

In ‘Towards a Poetics of Code’, Dexter lays out his view of code as a profoundly cultural practice. The locus of our study is not software-in-execution but software-in-creation, or source code. He writes:

‘Software is, certainly, much more than source code. It carries culture; it is Manevich’s metamedium. It tacitly and sometimes explicitly governs us; it is Lessig’s law. It is science and art and politics on scales grand and humble. Further, these myriad meanings of software cannot be read off the source code: they depend irreducibly on infrastructures of data, communication, politics, and affect. Just as a work of literature or a musical composition can be understood only at the most shallow level when read merely as a sequence of tokens, source code is only the source of meanings which emerge and erupt when the code entangles machines and minds.’

Dexter’s argument is that ***‘…code—source code—certainly has audiences for which it performs.’***

***The “recursive public” (Kelty) of free software is the most obvious contemporary example, though broad, trans-institutional collaboration has been a significant mode of software development since at least 1955 (Akera). Even Proprietary code, which by definition is potentially visible to fewer people than free software, nonetheless relies heavily on its audience. One of the many concerns of software engineering as an area of study and of practice is how to structure the developers on a project so that code is exposed to a critical audience as effectively yet efficiently as possible. One clear example is the practice of ‘pair programming,’ in which one developer writes code while the other looks on and strives to provide constructive critical feedback.***

***This concept of an audience which is invited, if not in fact expected, to modify the work being performed is integral to a broad swath of digital art; Kristie Fleckenstein takes this up under the rubric of “digital poetics.” She focuses particularly on the interactivity afforded by web-based publication of poems, especially forms of interaction which allow the “audience” to modify the work. “The phenomenon of deliberate interactivity raises intriguing questions concerning identity—ethos—and ethics: who is writing and who is taking responsibility for the writing?”***

* This takes me back to the last conference I contributed to here; on the responsibility of the writer.

***While not all source code is available to view by anyone interested (unlike the work of these digital poets), in general, anyone who is permitted to read source code, anyone in “the audience,” is potentially involved in modifying the work, perhaps as a co-developer, or as a code reviewer, or as a bug reporter. Thus, code has a much longer history of this kind of interactive digital poetics, although the author position in this domain may not have been problematized as rigorously. This gives rise to another complement of questions for this project: while the web, and perhaps free software in particular have provided and inspired new modes of interactive and collaborative reading and writing, what else can we learn about these modes by reading the history of software “authorship” through their lens? And, dually, what can we learn about the living history of software development by translating techniques and insights from literary analysis, such as Fleckenstein’s application of classical rhetoric, into the domain of source code?***

**The performativity of code - Linux**

In 1999 Linus Torvalds was the recipient of the prestigious Golden Nica prize awarded by Ars Electrionica in Austria. The jury of the .net category prize recognised that Torvalds however, was also ‘…representing all of those, who have worked on this project [Linux] in past years and will be participating in it in the future… It is also intended to spark a discussion about whether a source code itself can be an artwork.’

**Geoff Cox** - S*peaking Code*

The fullest treatment so far of code as something more than just its function is Geoff Cox’s *Speaking Code.* Here he argues that code is closer to speech than to writing and relates code to a collective speech act, one that is doubly articulated around expression and function: Quote ‘… the combination of formal description and creative action, what might be referred to as double-coding, is well established in software arts practice. …This exemplifies the material aspects of code both on the functional and the expressive level, …involving both formal logic and expressive aspects, its constraints and excesses.’

Coding is an embodied practice. ‘Like speech, program code is active in the world and has a lived body, indeed is intimately connected to a social body.” In this respect Cox understands code precisely as a performative in the Austinian sense, the one which Cramer says we should guard against. Cox goes further however. The subtitle of the book is Coding as Aesthetic and Political Expression. The latter claim is based on his view that not only do we make code speak, but code speaks us, pervading and formatting our actions and behaviours. Cox draws a parallel between this state of affairs and the way in which the act of interpellation in the Althusserian sense acts upon the body.

**A short digression - Permissions**

There is another digital literature concept that I want to put before you without going into it in any great detail and that concerns permission. As Sandy Baldwin points out:

*[The command] Chmod sets permissions to read, write and execute directories and files within a directory. To write. To create a file. To edit it. To delete it. A file is written only if permission is given. Web pages are no different. Every file is subject to permission. To read. To show the contents of a file. To see the name. A file is read only if permission is given. To execute. To execute a file. To run a program. A file is executed only if permission is given.*

In other words, it is impossible to carry out any written task on a computer without entering into permissible dialogue with it, without it allowing a dialogue to take place. This ties in with Geoff Cox’s notion of code as ‘speech act’ and a constant site of dialogue.

**Written dialogue within code: (Nick Montfort)**

Another site of dialogue within the digital apparatus is located within the code itself. Here, in general, two types of dialogues take place. One is between developers where a gloss or commentary is made on the compositional process of the code writing itself. This doesn’t impact on what the code can do but it gives a sense of the developmental process and underlines Scott Dexter’s point about the fundamental sociability and ongoing dialogic nature of code writing.

The second which is also a gloss is a exemplified in a Nick Montfort/Stephanie Stickland project, *The Sea and Spar Between*. Here the space of source code is used to set up a dialogue between the makers of the piece (Montfort and Strickland) and the user/reader (or whatever we are going to call this figure). This allows for not only instructions on how to access the work, but also contextualising and scholarly material on the source texts and an ongoing description of the processual thinking and compositional strategies that went into the work.

**Editing other people’s code - Taroko Gorge. http://nickm.com/taroko\_gorge/**

At this point, it would be interesting to show another of Nick Montfort’s projects, Taroko Gorge. It is exemplary in a number of ways. Firstly it is open source code, written by Montfort but available to everybody through creative commons. Having accessed the code, one can simply edit the variables and produce a variant on the original text. As you can see, there are a number of iterations of this work, and they are not restricted to English. This raises the question of whether this Polish version is a translation or not. And if it is, what is it a translation of? This question will arise again later on.

**Another short digression - Traumawien**

As a quick digression in this discussion about the relationship between writing and coding, coding as writing, etc. it would be worth making a brief mention of Traumawien. Traumawien is a Vienna-based publisher who “ …considers the paradox of transferring late-breaking digital aesthetics into book form, as new media narrative snapshots of literary genres otherwise quickly lost in the immense output produced by the web every second.” They are interested in preserving code within print culture rather than the output which is generated by the code itself.

**- Summary of the preceding argument/exposition.**

The major point I want to make in this first section of the paper then is that the digital writing which precedes the literary writing in a digital text work has to be understood and accounted for in its own right. Any full account of digital literary works will of necessity require analysis of the practices of code. What I have also shown I hope is that code is much more than simply a set of executable instructions. It is composed in a social, cultural and political context; it is composed as a set of dialogues; it is a site of dialogue; and it is open-ended, susceptible to being constantly edited and revised. Furthermore, under certain circumstances, it can be considered both an adjunct to creative practice and also a stand-alone creative practice.

So, if this covers the writing processes at work within the machine itself or rather at the hardware and software ends of the digital apparatus, the compositional and operational modes, we can turn our attention to the interface, to where the digital literary text is actually displayed. I want to begin by flagging up a problem for digital literature with Barthes’ model of authorship and readership as it is outline in ‘La Préparation du roman’

**Wreading and Wrighting - Editing and dialogue at the level of the interface:**

From the early days of theorising about digital literature, the term ‘reader’ has been problematic. It has long failed to adequately capture this function within the digital apparatus. In his 1997 book *Hypertext 2.0: The Convergence of Contemporary Critical Theory and Technology,* George Landow coined the term ‘wreader’ in relation to hypertext, as an amalgamation of writer and reader. The thinking behind this was to characterise hypertext as a new form of literary production where the writer and reader both engage in the creation of the text. Nothing new or radical here, of course. The reader has long been viewed as being deeply implicated in the creative process. And the idea of the ‘wreader’ is another manifestation of that. There is an equivalent in the term ‘prosumer’ which combines the idea of producer and consumer of a creative work. I have an aesthetic problem with the term,’wreader’, namely that it looks ugly – it has a sort of medieval air about it. I imagine a wreader as some sort of 14th century artisan basket weaver. However, there are two further shortcomings to the adoption of the term, ‘wreader’. Firstly, it firmly locates an engagement with digital text in the literary domain, creating the impression that the literary is the logical progenitor of digital writing. (In this context I am taking ‘literary’ to refer very roughly to a technology – that of print- and page-based media with its attendant reading strategies, theory of language, canon, etc.) This is hardly surprising in that Landow, as a Victorianist, is firmly rooted in the literary tradition, and that his particular area of interest is hypertext narrative, which is itself the area of digital writing most closely allied to the literary. Secondly, this term (and others like it) tacitly acknowledge the binary opposition of reader/writer and then simply seek to collapse that opposition linguistically. In other words, it looks like an integration of the two terms but no third term emerges from the integration. The words are agglutinated but the functions remain separated. This is contiguity rather integration. We write then read; rather than ‘in writing we are reading and in reading we are writing’.

While we’re in this neck of the woods, let me coin an equally medieval-sounding term, ‘wrighter’. A ‘wright’ in English is a ‘maker’, as in shipwright, wheelwright, etc. Wrighting then becomes a making – the term foregrounds the idea that digital text needs to be constructed/generated, rather than simply consumed. Does it make sense to think of the reader of a digital text as a maker? Insofar as any cultural activity could be called a making. One is reminded of Dante’s reference to poet, Arnaut Daniel as ‘il miglior fabbro’ the best maker. Perhaps we should find a place for the idea of the ‘digital textwright’ as a correlative to such terms as ‘wreader’ or ‘prosumer’.]

**Writer/readers in dialogue with each other.**

We have already seen in the example of Taroko Gorge an instance of collaborative editing of code and variables to create new but closely related digital texts. And I want to finish by looking at some other projects which take on board the potential of digital writing for dialogue and editing as a creative writing practice. The first of these is called the Reading Club and was devised and set up by Annie Abrahams and Emmanuel Guez. At each appearance of the Reading Club, four writers are invited to digitally edit in real time an existing text. The ‘performance’ lasts for twenty minutes and a video screen capture records the changes/editions that take place over the course of the performance.

[Show the video] <http://readingclub.fr/archive>

Interestingly, the four writers were physically separated from each other. Two were in Paris, one in Colorado and one in Puerto Rico. For me, one critical concept which emerged from this exercise was one of ‘respect’. To what extent should we respect the original and produce a recognisable version of it and to what extent should we bend it completely out of shape? In fact, the dynamic of this particular group was far less reverent than many of the others. [This may tie in with some of the remarks that Susan Barnosky will make later on the nature of translation] The other intriguing element of the exercise was the way in which as a writer you tuned in and out of what the other writers were doing. This felt like a sort of listening. At moments you were sharply focused on editing a particular section of the text and the next you were engrossed in listening to/reading what was going on around you. I also found that an agonistic impulse crept in - wanting to wade in and improve what others had written and yet also feeling uneasy about having ones own variations re-written.

**Texts in dialogue with each other in the digital environment**

If the Readers Project and Taroko Gorge constitute writing projects where writers are in dialogue with/editing code, and where writers are in dialogue with each other through the editing of text, there are also examples of digital literature where *texts* are in dialogue with each other.

This is Along the Briny Beach by J. R. Carpenter. <http://luckysoap.com/alongthebrinybeach/index.html> which builds on and expands the Taroko Gorge text generator

Here we see a number of texts scrolling against each other and the reader is given a modicum of control by being able to halt or reverse the progress of certain texts, or reveal texts against a contrasting background. The texts themselves are found and well known. In many ways these sort of works can be grouped together under the rubric of intertextuality, remix, remediation, mash-up, etc. all of which are recognised critical categories within literary analysis.

A more provocative example of texts in dialogue - in this case between digital and print-based texts - comes from John Cayley and Daniel Howe’s work, *How it is in the Commons Tongue*. Here Cayley and Howe wrote an algorithm which found strings of words on the Internet which when assembled reproduced exactly the Samuel Beckett text, *How it is*. They then printed the text in a hard copy which scrupulously referenced all the web sites from which they had extracted their fragments of text. They also sent a copy to the Beckett estate, who are notoriously aggressive in protecting the legacy of the great writer. Beckett famously said of Burroughs and Guysin’s cut-up technique “That’s not writing. It’s plumbing.” So I suspect the Cayley and Howe work would have had him spinning in his grave.

A variation on this project was a text I created with Kay Lovelace which was a mixture of original and found texts from which the reader can extract their ghostly doubles. [Explain the concept of The Fetch]

**In conclusion**.

I would say that as far as digital writing/digital literature is concerned, rather than having to establish its credentials as dialogic and editable, I would ask in what way is it ever *not* both of those. In my view this writing practice is an event rather than an object, a process not an outcome. It is a paradigm of an open-ended, multi-vocal performance which lasts for as long as it is in dialogue with its user/reader etc. and is then dispersed only to be re-assembled and come into being again at each moment of instantiation. To reiterate the point I made at the beginning of this survey, one of the things that makes digital literature interesting from our point of view here is the extent to which it encapsulates a processual and performative mode of writing which brings into sharp relief the notion of the visible and the invisible, the secret and the overt, the concealed and the revealed, the embodied and the ideal.

Digital literature bypassed Barthes. It would be interesting to know what he would have made of it..