

## Performing Software Quality

Neil McBride  
School of Computing  
De Montfort University  
The Gateway  
Leicester  
LE1 9BH

Email [nkm@dmu.ac.uk](mailto:nkm@dmu.ac.uk)  
Phone: 0116 207 8500

# Performing Software Quality

## Abstract

This paper explores the application of the research approaches developed by Norman Denzin to information systems. It explains the concepts of performance ethnography and autoethnography and examines the importance of storytelling, highlighting the importance of location, persons, action and emotion. The paper calls for a more personal approach to interpretive case studies. A more personal approach is illustrated using three pieces of autoethnography which look at software development practice.

## Introduction

***/\* alter and expand research approach and justification  
Look for more autoethnography. Put criticisms up front offer  
autoethnography as an alternative. \*/***

*“In the social sciences today there is no longer a God’s eye view that guarantees absolute methodological certainty. All inquiry reflects the standpoint of the inquirer. All observation is theory laden. There is no possibility of theory free or value-free knowledge. The days of naïve realism and naïve positivism are over.”* Denzin (2003, p 108)

Any research act involves an interpretation of phenomena and a presentation of that interpretation in a form that the audience can use to pick up the meaning. The interpretation is presented as a story. Whether that story is disguised in statistics or table, or presented as a case study, it still evokes meaning and transmits a concept or theory to the reader or listener.

Research is a process of constructing stories and recruiting an audience who will adopt the story, interpret it themselves, agree with the meaning and transmit it to others. Research is a value-chain of stories where researchers and listeners pass on stories in a continuous chain of transmission. These stories then form the basis of action for producing more stories in the form of further research, for changing the behaviour and activities of listeners, and for subduing and controlling the listeners’ surrounding environment through the production of artefacts (such as IT) and tools.

Whether the output of research is a theory of technology adoption or a theory of matter such as string theory, it is no more and no less than a story, an agreed interpretation, gaining the consensus of the audience about how things are. Many stories, which may be accepted as scientific fact are woven into the cultural consciousness and are part of the assumed knowledge base. As such they are still stories about the world we live in which may be

subsequently interpreted as being wrong and replaced by new interpretations, new stories. It is often the case that current stories need to be questioned and their influence disrupted. Disruption of current stories can only be achieved by the creation of new stories in the same way that Jesus' parables challenge existing assumptions and stories about the world and about moral truth.

If Denzin's comments are true, then any attempt to represent information systems as a positive science is bound to fail. Hence we need to explore a much wider range of approaches to representing phenomena, ideas, outcomes, frameworks, interactions in the information systems discipline. This paper explores one such approach, performance ethnography and applies it to the area of software quality, attempting to present stories and interpretations which may highlight issues in software quality, in particular pointing to the personal dimension in what is proposed as an engineering discipline.

## **What is Performance ethnography?**

Performance ethnography involves the presentation of the researcher's experience and interpretations in the form of poems, scripts, stories presented by actors on the stage or filmed. The focus moves from the printed page to the direct interaction at a location, at a point in time with an audience. The intention is to influence the audience with the outcome of invoking change. This adds a new layer of interpretation and interaction to the research process. The interpretation by performers and audience results in new understandings, which might not exist on the page. Denzin (2003, p247) suggests that the purpose of research is not just the production of new knowledge, but is pedagogical, political, moral and ethical. The research may be a teaching and learning experience, attempting to change the listener's mental model, to establish the moral position of the participants in the research and to highlight and challenge social inadequacies which have arisen from ingrained social and bureaucratic structures.

Much of performance ethnography involves the treatment of interview and other sources. They are rearranged and interpreted for performance. Interviews may be broken up into lines and stanzas to create a poetic effect which will invoke an emotional reaction. Dialogues and parts for voices are scripted from the interviews. At several levels, the performance of an interview or the researcher's interpretation of it, is a personal event. It is an outcome of reflecting on the experience. Furthermore, the interview itself can be viewed as a performance, in which the interviewer and interviewee interact, sharing interpretations. Hence Denzin recognised levels of performance in the interview and in the presentation of that interview as a performance. Within the interview and the researcher's reflections on the interview, Denzin seeks key points in the person's life, pivotal experiences which changed the person's attitudes and behaviour and may be remembered years later. Denzin (2001) calls these points epiphanies, to emphasize the drama and significance of the moment. A significant amount of Denzin's work, borne out of his experience of his father, has focussed on alcoholism, looking at people's experiences in Alcoholics Anonymous, exploring the point at which a

realisation and change occurs. His work in Performance Ethnography tackles issues of racism and culture, looking at how significant events in people's lives, affected their outlook years later.

## **What is autoethnography?**

A lot of Denzin's ethnographic studies involve reflections on his own experience, in his life and in his research. His studies into racism involve not only the presentation of the interview as a performable work, but also interspersed descriptions of his interaction with the subject, the environment of the subject, the exchanges with the subject after the interview. The researcher is not objectively distanced from the subject, but is part of the interaction. Other autoethnographic work involves descriptions of his relationship with his father, a discussion of racism and the treatment of native Americans cast within his experience of living in Rock Creek, Montana. They are stories about disruption and change. At one point Denzin tells the stories of the effect on him of the arrival of a rich woman from Las Vegas in Rock Creek and the effect on the natives. It is not clear why he tells the story, although it may be his experience of being a 'native' and what happens when lands are taken over by 'foreigners'.

Autoethnography recognises the importance of the researcher's experience. Since research is pedagogical and moral, involving the challenging of existing moral frameworks and the establishing of new moral frameworks, autoethnography is a good way of establishing points and encouraging the listener to be reflective. Often teachers use examples from their own lives, work experience and consultancy to illustrate points. Such autoethnographic telling of stories provides authenticity, establishes the teacher's moral standpoint and transmits underlying or tacit philosophical frameworks which may be key elements of the learning experience.

Autoethnography recognises that the research process itself is a learning experience, even if only in the way the research experiences and interprets the interview. The interview environment and experience will trigger ideas and connection with the researcher's experience. Autoethnography draws on the researcher's experience in a reflexive manner. The act of creating scripts and performing scripts may transform the material and draw out hidden lessons.

In this paper, I draw on my own experience as a software developer and systems analyst. Reflecting on what has happened to me and what I've caused to happen in the past leads to a different understanding of software quality and the process of achieving good software quality.

## **Telling Stories**

Stories and storytelling provides an almost primeval means of transmitting culture, establishing or creating norms and building socially acceptable behaviour patterns. Stories may shift understanding and challenge assumptions; they may encourage behaviour patterns or establish avoidance

behaviour acting like Belloc's cautionary tales. Take the following story, regularly told by engineers who maintain high speed centrifuges for biological research. These machines spin heavy titanium centrifuge heads at 37,000 or 59,000 rpm operating in a vacuum:

*In one London lab, the researcher put on a two hour run at 37,000 rpm. He didn't properly click the lid close. After an hour the vacuum leaked and the lid came open. The centrifuge head lifted out of the centrifuge, hovered in mid air and went through two walls before stopping!*

The effect of such a story is to connect with the listener's emotions. The dramatic image becomes fixed in one's mind, the emotion of fear is elicited and the listener always checks the lid.

Stories point out wrong behaviours, they highlight the contours of social injustices; they provide a trigger for change. They enable the questioning of prejudices, of patterns of working, of personal and social assumptions. Stories in people lives are told and retold, the epiphanies which become turning points are rehearsed as stories.

Stories also get the point across. The telling of a story will highlight the one specific concept, idea or problem which is central to the study. Walsham (2001, p81) tells the story of travelling with Geographical Information Systems scientists in India searching for an Institute by asking local people at the roadside and refusing to use a map. The reader can immediately understand why GIS technology may have difficulty in getting adopted within an Indian organisation.

At a social level, these stories behave like social memes, spreading through the social group, taking roots and establishing cultural meaning and norms. The key elements in the story may be conserved in transmission. Conversely, the story may get altered or mutated in transmission. The elements of the story which may be conserved include location, persons, action and emotion.

**Location:** Where the story takes place is important. In an interview, the location, the type of building, what the interviewee surrounds herself with is important. Setting out the location may elicit emotions and colour the action. Presenting the research as a performance may give the opportunity to portray a three dimensional location on stage.

**Persons:** Stories will centre around the activities of a person. Personal histories will enable an understanding of the person's identity. When Denzin interviews a 81-year-old woman who was involved in desegregation in the early 1960s, he describes her demeanour at the interview as "having the look of a patrician, a commanding presence, tall and graceful, but slow in her movements"(p96). Such descriptions set the 'person scene' so that they not an anonymised actor in an organisational case study but have presence and influence over the situation and the action. Actors may show the interviewees demeanour, attitude and even the way things are said, which will transmit more information and create greater understanding in the observer.

Action: The story must move. The actors take decisions, carry out purposeful activities, face the consequences of their decisions, and react to their environment. Such reaction then drives the story forward. Action, in terms of spoken dialogue or activities undertaken by the participants in research, may be better communicated by performance.

Emotion: At the heart of the story is emotion. Emotion and reaction to situations is often hidden or ignored in information systems case studies. The interviewees expose their feelings beyond the facts in the dialogue. They are acting emotionally within the system and are not simply neutral or analytical. The use of the information system may make them angry, worried, confident, or excited amongst a range of emotions. These emotions need to be captured and expressed. Performing the interview in some way may help to express emotions which are difficult to transmit just from the printed page. The actor can show the emotion. Eliciting emotions in the listener will increase the efficacy of the story and increase the pedagogical and moral impact.

## **Interpretive Case Studies**

Interpretive information systems research firstly recognises that the many participants in a study will have different views of an organisational event or phenomena and hence synthesises a presentation of the research – usually in the form of a case study – from those different views. Interpretive research recognises that there can be no one objective explanation of complex social phenomena. Secondly it recognises that the researcher views the material and the phenomena described through his or her own lens. Hence the written case study is really an interpretation of an interpretation. The lens of the participants and of the researcher filter, remove and focus the actual phenomena. There is, of course, a further layer of interpretation by the reader. While the contact between the participants and the researcher, usually by interview or observation is close and effective, the contact between the researcher and the audience is at a distance, mediated by a written paper. Performance ethnography attempts to close this gap by moving from written word to performance. However, there are other ways of closing the gap through the way the case study is written up, the development of what Denzin calls thick description, and the explicit presentation of the researcher's involvement in the research.

Even interpretive case studies in information systems may masquerade as more objective studies which they are not, by suggesting a distancing of the researcher from his or her subjects. Depersonalisation, objectivisation and structuring may occur.

## **Depersonalisation**

Participants are cast in organisational roles and the focus of the case study is on the organisation. People are seen only to the extent that they interact with the organisation or the information system. The pressing of particular theories on the case study such as actor-network theory or structuration

theory will not only treat the participant as faceless actors, but place them in groups such as systems developers, users, customers. Responses are too rapidly generalised and attempts to anonymise remove personalities. In particular any emotional depth and content is erased in an attempt to create some aura of generalisation and hence scientific validity. Furthermore people are quoted out of the context of who they are and the environment within which they work. Rich or thick descriptions will paint pictures of the person and their environment in the same way as a novel develops character and place.

## Objectivisation

In many case studies the researcher attempts to exclude themselves from the description. Any sense of the researcher's presence in an interview and of the performance that takes place is removed. The case study apparently describes something that is distant from the researcher, that the researcher did not influence. There is no recognition of the impact of the researcher's presence. Or of the way the background, philosophy, and emotions of the researcher colour and tone the resulting case study material.

## Structuring

Often a researcher will structure the story and material in the case study to lend itself towards a particular theoretical interpretation and to suggest an order whether is action, thought or motive that was not necessarily there. Stories do not follow the linearity and structure that some case studies suggest. Time might be compressed, participants talk about past and present in the same sentence. They range back and forth without linearity. The linearity is structured in by the researcher to give a false impression of order, of cause and effect that was not originally there. Hence the case study appears more objective and scientific than it really was. While the telling of any story is going to have some structure, it does not have to be linear. It can emerge from the participants' emotions and understanding of what happened.

Depersonalisation, objectivisation and structuring not only give a false veneer of scientific objectivity to the interpretive case study, they can also strip it of impact on the audience and of pedagogical value. To get a change in an audience, which may result in a change in organisational behaviour, the emotions must be engaged as well as the rational mind. This may be achieved by using different literary forms, using poems or dramatic dialogues, or if the research output must be limited to the written word, creating rich evocative descriptions, incorporating action and dialogue and creating an emotional layer. The researcher can describe her involvement through scene-setting, reflecting on her emotions and thoughts, and setting the phenomena within their life experiences. Some layer of auto-ethnography can be incorporated.

## **Performing Software Quality**

The three pieces presented below are my reflections on incidents and times in my life where software quality has been an issue or concern. So much of literature presents software quality as an engineering concern. Software quality arises from establishing procedures, from following standard processes, from measuring and recording. Hence if procedures are followed, the resulting software will be of a suitable quality. It seems to me this is not the whole story. Software writing may be seen more as an art. Hence the quality of the software will not only depend on the skill and artistry of the writer, and be affected by the writer's background, training and perhaps mentoring as well as the writer's natural ability, but it will also be a matter of subjective judgement by the perceiver rather than any objective measurement. Furthermore software quality may be affected by the quality of the relationships and interaction between the people creating it, and equally damaged by disrupted relationships and bad organisational support.

## A Sense of Excitement

During my time as a software consultant, I often walked over Blackfriars Bridge on my way to work. The morning walk across the Thames gave me time to think about the day's work and look into the Daily Express offices and reflect on the difference between what I did and what the journalists in the next door offices did:

A sense of excitement  
Elicited by the high of bitter coffee,  
The hum of a mainframe computer  
A small white giant  
Sitting behind my shoulder,  
Looking enquiringly.  
Lights flickering in anticipation of new input.  
Programs, data calls,  
Masses of SQL structured into poems across the page

We are building a system.  
There is a process of creation which occupies the mind  
On the train to Waterloo  
On the walk across Blackfriars Bridge  
The quiet river  
The neon gaudy lights of the newspaper offices.  
The blank walls of the system development house.

Who is more creative?  
The news paper writers in their canteen looking across the river,  
Writing the headlines. Structuring stories..  
Or the programmers on the second floor,  
Working on reference tables  
Creating a model of the world  
Encased in the language of systems?  
Communicating stories of patients' diseases  
Movement between hospitals and wards and theatres.

Procedures and tests.  
Episodes of care.

In the newspaper the editor judges the quality of stories.  
A process of reading brings an intuitive view  
Good, bad, too wordy.  
Not measurements and numbers  
But judgement based on feel.

Two doors up, I examine the aesthetics of a program.  
The feel of the text,  
The structuring of calls,  
The flow of the model of medical care.  
It sounds good. It feels right.  
And in the absence of the obvious programming bugs  
It's a good program.

## The Visit

Part of my software consultancy work involved developing systems for hospitals. As more hospitals took on the systems we were building, quality became a big issue such that we employed a quality manager. This piece, presented as if I was being interviewed, reflects on the clash between formal quality approaches and the business demands of a commercial software project:

We were working hard on a new version of the software product for Dublin  
There were lots of new programs to be written.  
The alterations required by the client were so many that it was like writing a new system.  
I don't know how long was left, but it wasn't long.  
I wanted the client to move the deadline.  
My boss wouldn't hear of it.  
We knew the links with other hospital systems weren't going to be ready for six months.  
The hospital staff wouldn't be ready for the system until way into the New Year.  
And yet we had to deliver it by Christmas.  
Some of the new programs weren't written.  
Lots weren't complete.

I knew what everybody was doing.  
We'd had weekly meetings to check on progress.  
I'd sit down with the others and check through their programs  
What database calls was it making?  
How did it match the specification?  
Then after debugging, if we were happy with the results of tests, the program would be put in the main repository.

The lead contractor was a well-known hardware manufacturer.  
We effectively worked for the lead contractor, not the hospital.  
We were told the contractor was coming over from Dublin.  
She would question us about the state of the project.  
She wanted to see the quality plan.

We didn't have a quality plan.  
We worked as a team, checking each other's programs.  
We could pick up each others programs and work on them.  
But we didn't have a quality plan.

I had to write one before she came.  
I thought of what we did informally and naturally.  
I could write it up as a formal process.

Walkthroughs.  
I called them inspections.  
They weren't really.  
We didn't have the people to appoint the formal roles you find in Fagan's inspections.  
Not did we have a defects database.  
But we did follow through errors  
And record quality history at the front of the program.  
I wrote up the steps  
And showed it to people.  
Is this what we really do?  
I guess so, they said.  
I defined the records we kept and listed the information recorded at the start of the program.  
I recorded our controls on the repository  
Which meant that two people couldn't take the same program out at once.  
I had the projects quality manual bound in blue with a title page and contents.  
It looked quite good.  
It said what we did.  
And we made sure we did it.  
There was a record for each program.

No program was released into the repository unless it had had a walkthrough.  
The defects corrected.  
The quality process, or so we called it, signed off.  
It said what we would stick to.  
It would guarantee a quality system  
Day by day we would build up quality records

When the account manager arrived from Dublin.  
I showed her the quality manual.  
She seemed quite impressed.  
It was what the hospital would expect  
We should record the results of every walkthrough.  
I assured her no program would be released with out a quality record.

But it wasn't what was on her mind.  
She questioned me about the team meetings.  
About work completed, work to be done.  
How many programs were ready?  
I didn't know. It wasn't my job to recruit more people.  
Anyway, would it work at this late stage to add more programmers?  
You can't train up people quickly.

But more people means getting the project out quicker.  
She talked about the deadline many times.  
It has to be in by Christmas.  
All programs had to be written by then.  
We couldn't just provide some main functions.  
Every function in the agreed requirements specification..  
She mentioned agreed.  
I hadn't agreed to it.  
It was unrealistic.  
I wanted quality programs.  
I wanted to use the quality plan she had asked me to write.

A storm was brewing  
In the afternoon fifty mile an hour winds battered the second floor windows.  
We could see signs swing about and lights flicker.  
We worried about the protected power source.  
We did an extra backup.

Two weeks before Christmas  
Lots of programs hadn't been walked-through.  
There were bugs everywhere.  
The boss said you release it this Friday.  
I argued we could wait till after Christmas.  
The hospital wouldn't use it before Christmas.  
That would at least give us time to work on episodes of care.

That was the main program  
There's always a main program.  
Where all the processing, all the main business processes occur.  
It was critical.  
We'd all worked on it.  
It still had bugs.

We needed more time for episodes of care.  
The boss said that's the way it is.  
The programs have got to be with the hospital by Friday.  
We worked up to the last moments of Thursday.

As the motorcycle courier waited downstairs.  
We copied the latest versions of the programs to tape.  
I packed it in a Jiffy bag

And ran downstairs to hand it to the courier.  
It was at Heathrow by five o'clock.

Tomorrow two programmers would fly to Dublin  
And continue programming.  
And debugging.  
In front of the client.  
Few of the walkthroughs had been done.  
The defects database was never built.

But the project was delivered on time.  
The account manager got a Christmas bonus.

## IBM Hursley, 1977

During my undergraduate years, I worked as a programmer at IBM's research laboratories, working on various research projects. In my final year, I worked on systems programming for new hardware developments. At the end of my three month stay I would write a report. This piece uses my original report as a basis for reflecting on the software development process. Extracts from the original document are followed by reflection and comment:

*Having successfully completed my second year reading Microbiology at Queen Elizabeth College, London; I commenced my third period of employment at Hursley as a vacation student on June 27 in the RAS (Reliability and Serviceability) group of Distributed Display Products.*

Cycling from Southampton to Hursley Park  
Across a savannah landscape  
The parched fields shimmering in the heat,  
The dry grass turned to orange, red and yellow,  
Giraffes striding between the tall browned trees.  
Lions hiding in the white grass  
African plain painted over English countryside  
I chase a bus two stops to Hursley.  
Past the colonial red post box  
Into the grounds of Hursley Park.

The imposing columns of Hursley House.  
The scent of new carpet  
Display cases containing handwoven memory  
Winding staircases to servant quarters  
Now offices with views across the park.  
Large new buildings, linked by glass corridors  
Where 370s resided carefully tended by an army of operators.  
Tealadies with trolleys.  
Coffee, biscuits, cakes.  
Working at the 3270 terminal with the word 'IBM' burnt onto the CRT.

## ***Invalid Operation Codes***

*The Motorola M6800 microprocessor is typical of most microprocessors currently available. It consists of an integrated circuit chip in a ceramic or plastic case 50mm by 40 mm. Input and output are through 40 pins, allowing data and operation codes to be fed in, and data outputted. It has an instruction set of 72 instructions each of which can have up to four addressing modes. Programming is done using mnemonics and is assembled into object code, consisting of hexadecimal operation sequences, by the assembler. My first task was to write a program which fed invalid operation codes into the MPU and checked how the MPU reacted. This task involved writing several backup subroutines in order to display messages indicating the affect (sic) of the individual operation code.*

The IBM8100, as it was to become, was an early project to put processing power into a dumb 3270 terminal. It was a confidential project called Hotspur then. Documentation was written on paper with red lines across it and the words 'IBM Confidential' in the middle of the page. Once when I requested a document, a security man had to stand outside our office while I read it.

Running assembler programs was a difficult task. I knew assembler from university where I did an assembler course alongside organic chemistry and medical microbiology. The assembler there was on an Interdata machine. I almost went to work for them in Middlesex.

There was an M6800 simulator on the 370 mainframe. Having written the assembler code on paper, I would search for and queue for a 3720 terminal to type it in and test it on the simulator. The simulator could show what would happen with correct operation codes but not invalid codes. This had to be tested on the test rig. I would have to book time, download the programs on to a tape. The actual test rig was behind two set of doors. I'd knock to get an engineer to let me in. I'd load the tape on to a 1130 which acted as a host for the test 3270 which had the microprocessor in it. Then I'd run it by pressing some keys on the 3270 and get some output to study. I'd sit next to the electronic engineer, the smell of soldering irons in the air.

This programming was experimenting to find out what Motorola may know, but IBM didn't. There were patterns as to how invalid operation codes were treated.

I think programming must start by exploring the environment. What happens when I write a particular instruction? What goes wrong? What works? How does it look? It involves trying things out like my son tries out combinations of Lego bricks. It involves experimenting like throwing pottery, like trying out hundreds of bagless vacuum cleaner prototypes. Like a chimpanzee experimenting with sticks. The sticks for me were programs running on the M6800 simulator with which I probed the real microprocessor. I don't think you can formalise such a design process, you need room and time. You need to fail and watch things breakdown. I learnt to handle the processor, to understand its behaviour.

## ***Reorganising the old diagnostic program***

*It was at this point that I picked up the existing diagnostic program. This consisted of main line code which tested non-storage followed by storage instructions. At apparently arbitrary points it branched off to eight and sixteen bit divide and sixteen bit multiple routines, on each for storage and non storage instructions; and it finally executed a hexadecimal conversion routine. It has a register exerciser as an interrupt handler.*

A lot of a programmer's work consists of taking things others have written and trying to make sense of them. What I was given in the old diagnostics program didn't make sense. It was an old-style monolithic program. I was brought up on structured programming. In order to fit it into my mental model I just had to rearrange it. I made twelve independent diagnostic subroutines out of the one program.

There was a lot of extension work to do as well. I added new RAM testing routines, a ROM tester, data bus exerciser, interrupt handlers.

Really it's a matter of taking ownership, of turning someone else's work into one's own. As such the work either bends to one's will and becomes what you want it to be, or it breaks. Writing any software is a creative process. It's really creating an artwork with aesthetics as well as structure. For me the art in software was in order, in almost geometrically structures.

Gradually I took over what was left of the program. The original, non-aesthetic, monolithic program atrophied. What emerged was my creation, my artwork.

### ***Rewriting the core MPU diagnostics***

*Examination of the statistics on MPU usage for the reorganised old diagnostic program would not confirm the validity of the divide and multiply routines as MPU checking tools. Doubts had also been expressed by people outside RAS as to their usefulness. It was eventually decided that the amount of testing achieved did not justify the space they took up. Also there was an annoying lack of order in the instruction testing routines within the old diagnostics.*

I think I was pleased when there was some rational justification for replacing the last part of the original program with my own program. Although a lot of experimentation was needed to do this. Testing every condition code setting of every instruction proved too costly in space. I think the experiments meant I understood more of why the previous programmer has taken his particular approach. There was lots of rewriting and thinking and rewriting. But unlike much programming there was a hard metric of success. Statistics could be examined to see how well the program exercised the code. My new programs show a considerable improvement.

It was a creative process. I was the author of a program whose organisation and structure reflected my way of thinking, my view of the problem, my way of writing. It was not a case of software engineering but software creation.

Towards the end of my time a senior lab manager visited. We all crowded into the test lab and the loaded 3270 with its M6800 was turned on. Nothing happened for a second, then logon screens came up. It had to be pointed out to the manager that in that second my diagnostic routines were testing and

exercising every part of the hardware. I think there was a sense of achievement in knowing that anytime someone switched on an IBM8100, anywhere, my small creation would be the first thing to run.

### **Quality Meetings**

*During my stay I attended several meetings including the I0 and I1 for another group (these are basically design reviews) and the I0 for our own code.*

All my work during the summer of 1977 was done without formal quality assurance processes. Requirements, activities, programs and problems were discussed with my manager and senior managers. It was up to me to think about quality, which I felt naturally came out of structuring the programs, using routines to break down code into meaningful sections and adding structured comments as I went along.

I'd come across Fagan's Inspections (Fagan, 1976), invented by Michael Fagan at IBM and thought it was an interesting approach. Despite the informality of my approach, I was interested in formal approaches to quality. I found out that another group outside RAS held a sort of inspection, called design meetings. I got my manager to let me attend their meetings. I then persuaded my manager that a design meeting would be a good idea. I got a room booked and got colleagues in to review the design for the diagnostics programs.

### **I0 Meeting**

My program's OK.  
There were minor bugs.  
But that's what the meeting's about.  
I feel reassured,  
Accepted.  
Part of the group.  
The I0 is a social meeting  
A software bar mitzvah  
A congratulatory pat-on-the-back.  
I have created a rite of passage  
I have become visible and known  
My work has passed I0  
Like some kind of examination  
Which sets social status in a troop of programmers.

\*

Sitting in a spare office,  
My feet on a desk  
Talking to a friend  
I feel at peace, looking across the fields.

White birds fly across the scorched fields

Like cattle egrets searching for refreshing pools.  
Further still the glaring sun reflects off a hundred car roofs

\*

On the last Sunday before returning to college, there was a party at a manager's house. We had drinks. I talked with people. I don't remember anything of what was said. I'd received a special contribution award. When I got home my mother said, 'You could have stayed with us. It was your last Sunday before going back to London' She was upset. I felt bad. Looking back I feel I let her down. She was more important than meetings, design reviews and IBM. Perhaps at the heart of quality are relationships and trust and sharing and honesty. The next day I was on the train back to college.

## Conclusions

The work of Denzin (2001, 2003) is in the mould of critical research. In dealing with the cultural issues that concern him – alcoholism and racism - his aim is to challenge and change the institutional structures. His social science, which he describes as autoethnographic, vulnerable, performative and critical (2003, p105), is one where the researcher's efforts cause learning in society which results in change. He recognises that this change results from the way the consumers of the research change their interpretations of the institutes and social phenomena around them. He also recognised that this change is not just, or primary, a change in rational thought and models, but a change in emotions and motivation. Therefore, in pursuing performance, he is looking for ways of engaging the emotions. He also recognises the thread of performance which permeates the whole research process. The social subject is a performer in his environment. The interviewing and questioning process is a performance about a performance. That further is interpreted by the research as a work which should be performed to engage the senses and the emotions. Good social research should be manifested as plays, poems and novels, which engage the audience, create a pedagogical effect and lead to a research outcome manifested in a change of behaviour. In the field of race studies, it could be suggested that Harper Lee's 'To Kill a Mockingbird' was a much more effective piece of social research than any number of surveys. In the field of software engineering, Fred Brook's 'No Silver Bullet' may be much more effective than any quantitative survey into software failures. Hence, I would suggest that interpretive approaches to information systems research need to be more personal, engaging more with the subject and audience. Thick descriptions should be developed. Case studies should be performed and different ways of presenting material developed which engage with the senses and emotions of the audience. Researchers need to develop more creative approaches to case study papers. The feedback from initial performances of 'A Sense of Excitement' and 'The Visit' in student classes suggested that the message concerning the personal and people nature came across strongly and was a surprise to some students. A performance may also tease out unintended or unexpected

messages which may otherwise remain embedded in the text. In the case of the student class performances, a nostalgia for the straightforward days of mainframe computing and the dominance of technical skills came across. The area of study I have addressed in this paper is software quality. Texts and journals on software quality portray the discipline as a formal engineering discipline, where quality software emerges from well controlled engineering processes. Schulmeyer and McManus (1999) view software quality assurance as an engineering process, with inputs, outputs and measurables, where quality results partly from the elimination of human irrationality and the application of methods, processes and metrics. An examination of the contents of the Software Quality Journal for the last four years revealed nothing addressing the human management and emotional issues behind software quality.

I would suggest that software development is more of an art than an engineering discipline. Software quality is a product of relationships. It depends on the quality of the relationships between supplier and customer. It emerges from the effects of the innate skills of the programmer, the interactions and relationships amongst the team and the management environment. Processes, procedures, standards and documentation cannot guarantee quality software. Software quality processes may be emergent, developing as the team's thinking and maturity develop.

Like many areas of information systems research, there is a need to establish the primacy of social relationships and human interaction in the effectiveness of the IT artefact. Approaches like performance ethnography may help in providing different ways of teaching people and eliciting behavioural changes as a product of research.

## References

- Denzin, N. (2001) *Interpretive Interactionism* Sage California.  
Denzin, N. (2003) *Performance Ethnography* Sage, California.  
Fagan, M.E. (1976) Design and Code Inspections to Reduce Errors in Program Development. *IBM Systems Journal*, 15(3) 182-211  
Schulmeyer, G.G. and McManus, J.I. (1999) *Handbook of Software Quality Assurance*. Prentice Hall, New Jersey.  
Walsham, G. (2001) *Making a World of Difference: IT in a Global Context*.