Project Wavell – a memorandum

Overview

The Mission – to develop a computerised Command and Control system for the British Army. The theory was that the increased effectiveness was greater value-for-money than buying more tanks!

The Task Force – the main Contractor was Plessey Radar. This may sound like an unlikely choice, there being no radar in the project; but Plessey Radar had not long before made their reputation by handing over a very large software project (Project Linesman for the RAF), in time and in budget. The present writer knows of no other MOD software project that ended so well. Many of the Wavell team had been involved in Linesman. But Plessey had recruited a retired Army Major, Barry Moore, to advise the team on Army working practices and to act as liaison; he told us many stories from his service in Northern Ireland.

The project had three main components;-

- The hardware
- The lower level software including communications (led by the writer)
- The higher-level "applications" software to provide the command-and-control capability (led by Duncan Scott)

The software team was based in Stoke Park House, Stoke Poges (as in Gray's "Elegy"); the house sits in the middle of a golf course (made famous in the film "Goldfinger") – while the members played golf they had no idea that classified software projects were being put together in the basements and attics of their clubhouse. But we could wander around the course during our lunch-breaks, which was fun when a PGA championship was held at Stoke Poges.

The timescale – Planning had been happening during the early 1970's, and design work on the software started in 1975. The hardware and base software was acceptance-tested at Blandford Camp in 1978. The whole system was tested in one company of BAOR Germany in approximately 1980 (my future daughter-in-law's brother was involved in that, I discovered many years later).

The Hardware

It seems a simple concept – put a computer in the back of a land-rover and off you go. The problems were associated with the fact that the land-rover would drive up steep slopes, into ditches and over tree roots, never mind being in a potential hostile military situation. Vibration is the enemy of computer reliability, and to a magnetic hard disk of the 1970's it was "death". So every component of the system had to be "ruggedized", and so secured into the land-rover that while on the move nothing would be damaged. It was not intended that the computers would "run" while on the move – each land-rover would move to a new location, ideally hidden in a barn or under a clump of trees, set up, start up the computers and establish communications, and go operational for a period; then it would shut down and move to a new location.

The computers were by Digital Equipment Corporation (DEC) and were PDP-11's. The computer printed circuit boards, normally easily mounted, were encased in aluminium frames – processor, memory and communications boards. The power supplies needed extra work, and the hard disk had

to have a special case built for it – if memory serves me right, the hard disk box sat in the front passenger seat! Appendix 2 warns us how sensitive hard disks are.

The "user interface" comprised a keyboard and printer (ruggedized) and VDU's for the command and control system (ruggedised and TEMPEST-proofed[1]).

The communications interface was a DEC comms board(s); this connected directly to Army BRUIN units that provided encryption over radio links. This gave a digital signal to adjacent computers, of typically 9,600bps (which seems ridiculously slow by modern standards, but was quite good in the mid-1900's; typical speeds at that time were 2,400, 4,800 or 9,600 bps).

The base-level software

The operating system was DEC's RSX-11M/M-plus software, the standard for the PDP-11 computers. No modification was needed (though I did find a "bug" in it that DEC fixed almost immediately). All that was needed to add to this was a set of "batch" command files, so that a single command typed on the keyboard would do everything needed for (for example) start-up or shut-down.

Development Aids were written; these were tools to help the programmers test their software .. they were largely based on the equivalent tools used for the "Linesman" project (written by the same author) but adapted for the RSX-11 operating system.

The communications software presented a major challenge. DEC was developing its own software called DECnet; at the time this was "Phase 1", which allowed a single link to be used – in other words it linked just 2 computers together. It was planned that in time it would be extended to cover a network of computers, but nobody in DEC UK had any idea how this would be done or when; so in February 1977 Barry Moore and I flew to the DEC headquarters in Maynard MA to discover what the plans were. Sadly, the result was that there was no timescale and no firm plan … we got the impression that no one there really knew how to do it! So we returned to the UK, and I started to design a computer network from first principles.

In Appendix A I have described the design. It was as far as I can tell the world's first operational fully-routing computer network. As a defence contractor, intellectual property for everything we did belonged to the MOD; I as an employee had no personal rights. I understand that the MOD passed on the design to the USA, which helped them implement a number of defence and defence contractor networks (the prototype plans came from BBN in the USA in 1979) which eventually spawned what we now know as the Internet.

We had a set of non-ruggedised PDP-11's in the basement at Stoke Park, and there I tested the design, adapting it as I discovered problems. It needed something to generate test messages, and I thought the easiest test would be to send 1-line test messages. Basically, the operator typed in the code for the destination computer (or "ALL"), and the rest of the line was his message, e.g.

HQ1 All set up and ready to go operational

and at HQ1 (or every connected computer) the teletype would print the time, the code of the sending computer and the message, e.g.

11:43 FR2 All set up and ready to go operational

[1] TEMPEST was a way of safeguarding information against eaves-dropping; the Cathode Ray tube technology of the time emitted radio signals that could be picked up remotely.

This may be the world's first rudimentary email system!

Acceptance testing happened at Blandford Camp in Dorset; the ruggedized hardware had been fitted to the land-rovers, so all that was needed was the test software on a disk pack, one per land-rover. The Plessey team set up their HQ at a pub in Pimperne, where we had done a deal for B&B and an evening meal for the team. Each morning we came into the Camp, saw the troops set off to do their exercises, awaited their return, and then asked them how it had gone … "Fine" was the typical response. We tried to get improvement suggestions, but none came – it worked as it had been intended, there were no hardware or software issues.

I have to say that the "pseudo email" test software took a hammering during the tests. Up above I've described the sort of messages I expected the Army to use for testing. The ones actually used I will not record, but fell mainly into two categories (a) complaints at the unfairness of the world, weather, officers, and equipment, and (b) lurid descriptions of what soldier A planned to do by, with, to or for the blonde girl in the NAAFI, and incredulity on behalf of his fellow-soldiers at her willingness to be within 1000 yards of A, or his physical ability to perform any of the stated actions. Enough said! But the simplicity of use of this system meant that the troops took to it like proverbial ducks to water.

I understand that the tests (devised by the MOD) included

- Communicating in poor signal conditions (pull down the aerials to the point where speech is totally garbled, and see if data and get through; this was incidentally a major positive, because visible aerials meant high risk of detection, which meant high risk of an incoming missile),
- Ensuring that messages from A to D via B and C got through in those conditions, without corruption, and
- Ensuring that the equipment and software could be started up, run, shutdown as planned.

The Application Software

I was not really involved in this, other than having a general idea of what was needed; indeed, once the Blandford testing was complete, and I'd finished writing up all the documentation, I left Plessey and joined DEC UK. So what I write here is much more vague.

The general concept here is that of a shared map and a shared database of items of interest. During computer start-up, the latter would be sent to the newly-started computer, so it was quickly up to date; as information was added, it would be copied around the network. The general idea was that everyone would know the same information about movements (us and the enemy), targets and so on.

The application software was acceptance tested by BAOR in Germany. One of the 4 companies (I think "D" coy) was equipped with Wavell. All 4 companies were set exercises to do; the Wavell company completed all their tasks in record time, and in complete voice radio silence. All of a sudden, Wavell stopped being a weird experiment, and became the Army's top priority.

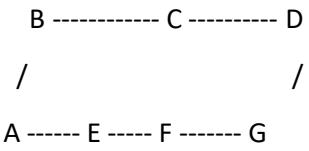
Bernard J Harris

Embsay, North Yorkshire; March 2023

Appendix 1 : The communications software design

This is highly simplified!

To "route" messages around a network, each computer needs a "memory map" of which computers are connected to which others; it then clearly needs to be able to calculate the best route from A to D. A picture may be helpful …



Creating the "memory map" What we start with is a list of the physical links coming out of "this" computer. So we then start to send test messages out on each link … "who are you, I am C"; when another computer (say "B") receives such a message on his link 2, he knows that he is connected to C on link 2, so he (1) updates his table "B is connected to C", (2) sends an acknowledgement back on link 2 … "thank you C, I am B", and (3) sends out a message on all other links to say "B connects to C". All these messages are of course in codes, not written English!

On receipt of such a message every other computer updates their table and passes the message on to all their neighbours.

Detecting a link going down Whenever C sends a message to B from now on, the software will get a signal back "delivered" or "not delivered"; if a "not delivered" comes back with a reason "no one there", he removes the B-C connection from his memory map, and tells all his neighbours that B and C are no longer connected; The neighbours update their tables and pass it on.

To detect a link going down when there are no operational messages to send, we introduce a timer – if no operational message has been sent or received for **X** seconds, send a test message.

Problems with the above The astute reader will rapidly work out that what is described is calamitous; (a) because these messages will go on for ever, (b) because there is a risk that the B-C link is intermittent (e.g. through loss of signal or during cable-plugging). To solve these we introduced two rules; (a) if you get a message (link B-C is up/down) and you already know that, you ignore the message and don't pass it on – this stops endless transmission of messages; (b) if another computer (say A) tells you the state of one of your links, and you know it to be false, you then broadcast the true state of that link; owing to rule a, it won't need to go too far before it expires.

Calculating the best route With the memory map in place, the computer can work out if there is a route to the destination, and if so, which of "my" links is the best one to send it to. The message may come from "my" computer's application software, or it might have come in from elsewhere. In the internet of today, the speed and capacity of a link will be factors; with Wavell they were all the same so I didn't need to factor these things into the equation … the best route was always the shortest number of "hops".

Delivery undistorted and in order It is important that messages are passed on in the correct order, otherwise the application database may be a false picture. So each message is numbered; along any link, such as B-C, one message may be corrupted in transmission, so all the time there is a "handshake" saying "I've received up to 123 OK but need 124", so the sending computer can get rid of all messages up to 123 and re-send 124. The receiving computer must hold on to 125 and 126

until 124 is safely received, but can then pass on 124, 125 and 126 and tell the sending computer that messages up to 126 have been safely received.

At application level, there was a similar message number, so that sequence could be preserved end-to-end as well as over the links.

Appendix 2 : References

National Archives : ref DEFE 72/201 : Project WAVELL, automatic data processing system for HQ 1

British Army - The Royal Signals - Other Communications Systems -Wavell - Slim - Bates - Vixen - Scimitar - Jaguar - NCRS - Mould - Brahms - Dust - Army Fixed Telecommunications Systems - Armed Forces - a9a9a

Full article: Project Spaceman: early British computer security and automatic data processing (tandfonline.com)

Project WAVELL - NASA/ADS (harvard.edu) Written by Russell Fairchild

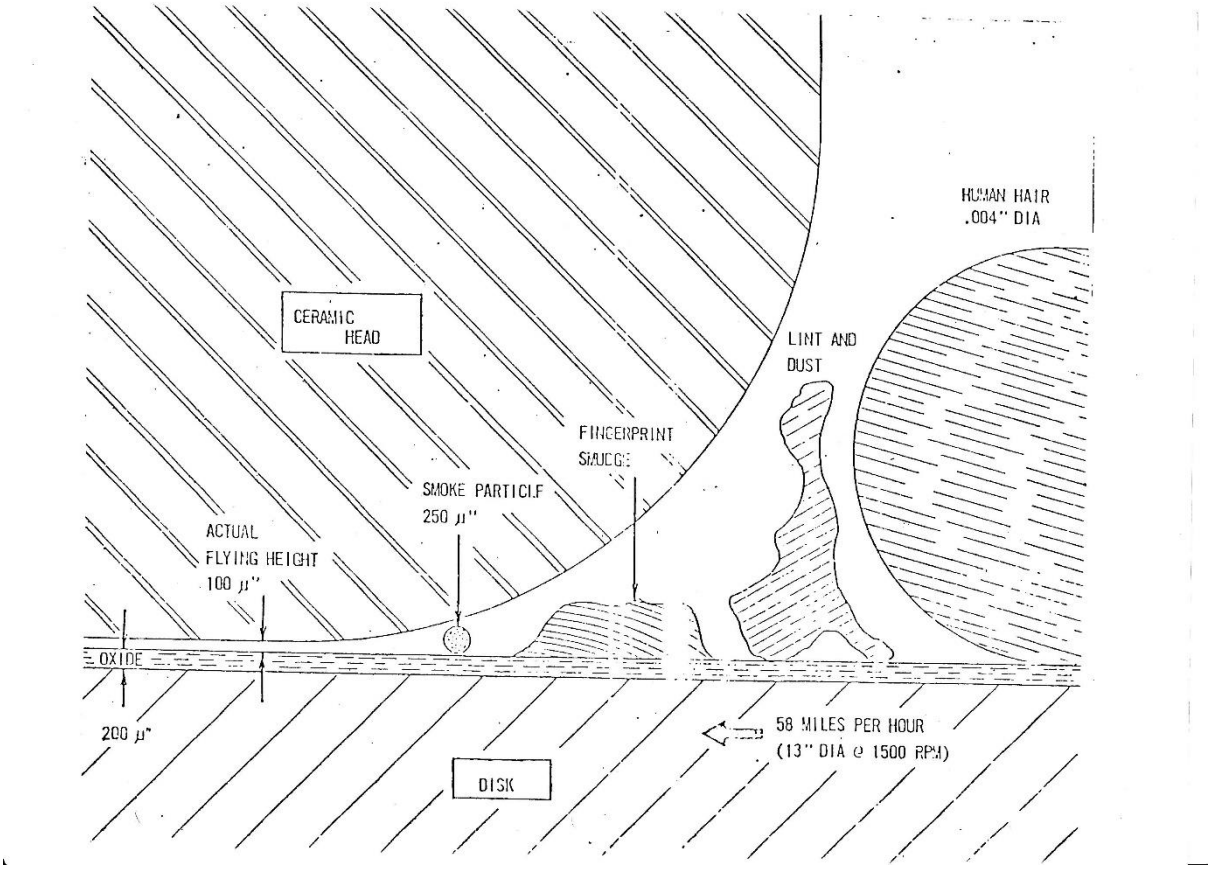
Military and Technology (linkedin.com) Describes how Army staff reacted to computers!

Appendix 2 : Hard disk tolerances

This picture was in common circulation in the 1970's, and was a warning to everyone about the danger of damaging hard disks. Should the floating disk head (on the left) hit an obstruction, it would be forced upwards, but then it would crash downwards afterwards, making a dent in the oxide coating and ruining the disk.

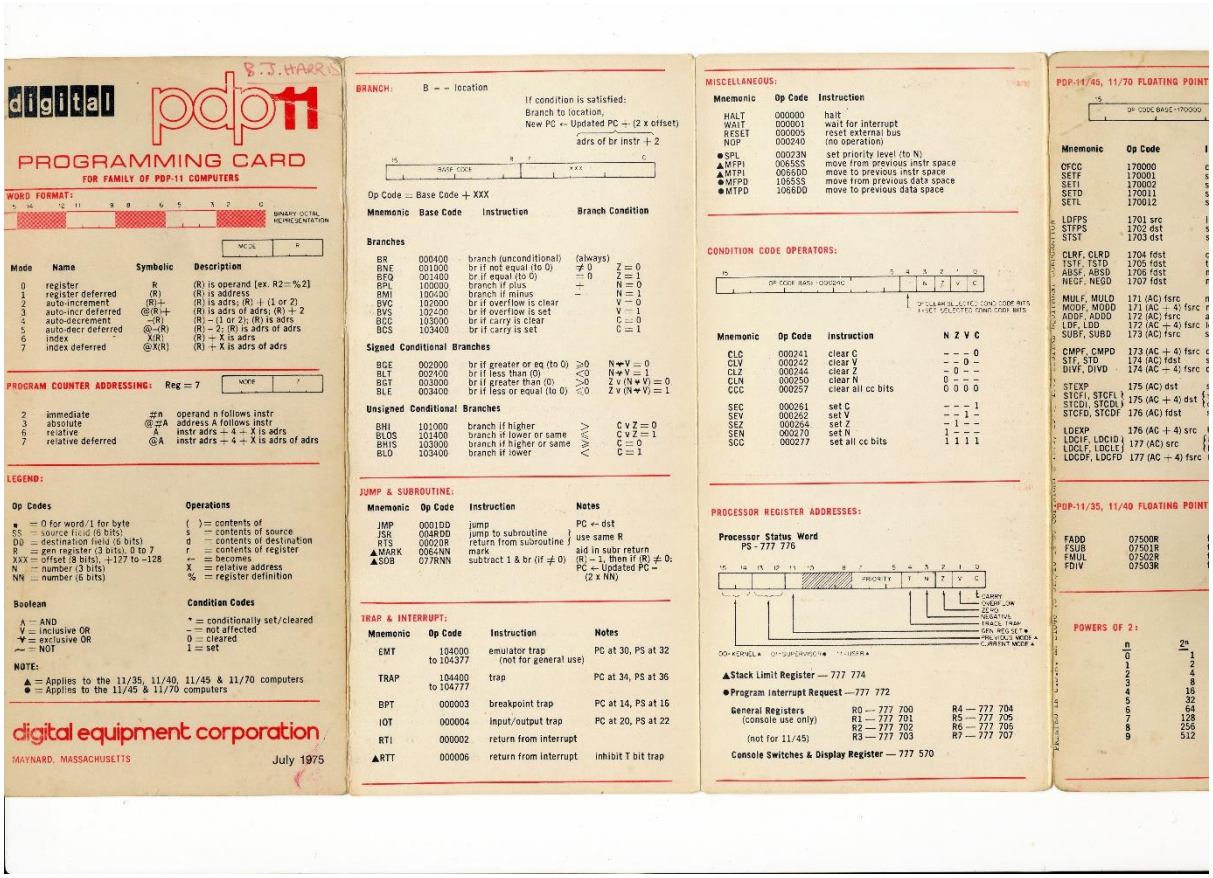
The "obstacles" from left to right are ;-

- A smoke particle
- A fingerprint smudge
- Lint or duct
- A human hair



Appendix 3 : PDP-11 programming card

This is about a quarter of a programming card for the DEC PDP-11 computer, and tells the programmer “all you need to know” about how the computer works. The original has 10 “sides, was folded in concertina fashion, and slipped into a shirt pocket. This one was mine … if it looks somewhat grubby, that’s because it was very well used!



<ends>