THE DESIGN OF A NETWORK COMMAND LANGUAGE

PHILLIP CAMPBELL JENKINS

1980

A thesis presented in partial fulfilment of the requirements for the degree of Master of Arts in Computer Science at Massey University.

ABSTRACT

Most computer series have their own distinct operating system control language. At present there is no world-wide standard control language, but the recent development of heterogeneous networks, where users may access many different computers, has highlighted the need for one.

A project to set up an experimental network, called KIWINET, between Massey and Victoria Universities, initiated research into the design of a standard control language for the network. The result of this research is a high-level, block structured language called the Network Command Language (NCL). The design of this language and its implementation are discussed.

ACKNOWLEDGEMENTS

The initial work for this thesis was done while I was employed by the Massey University Computer Centre. I would like to thank all the staff of the Computer Centre and also the staff of the Computer Science Departments of Massey and Victoria Universities, for their continued support of the project, both as technical advisers and as constructive voices during discussions. In particular I would like to thank Mr Neil James, of Massey University Computer Centre, and Mr Keith Hopper, now of Leeds University, England, who have both been, at various time, my supervisors and who have been intimately concerned with the design of NCL.

Special thanks are owed to my present supervisor, Mr Lindsay Groves, of Massey University Computer Science Department, and his colleague Mr Tom Docker, for many useful discussions and for their ever helpful criticism.

I would also like to acknowledge the invaluable constructive criticism and suggestions concerning all phases of the design of NCL, received from Professor Unger and his colleagues of Dortmund University, West Germany.

Lastly, I would like to thank Mrs Thomson, for doing the proofreading, and my fiancée, Alison, for making even the bad times good.

CONTENTS

<><><><><><><><><><><>

TABLE OF DIAGRAMS

CHAPTER 1

INTRODUCTION

"About $\$10^9$/year are lost due to command language errors."

T. B. Steel

Said during discussion at the IFIP

Working Conference on Command Languages,

Sweden, 1974.

1.1 The emergence of networks

In 1968 the Advanced Research Projects Agency (ARPA) of the U.S. Department of Defense began to create what is now called ARPANET; a network of about 60 heterogeneous computers at geographically distributed sites throughout the United States. It includes satellite links to computers in Hawaii, Norway and London.

The main idea behind ARPANET was to allow the full resources of the computers in the network to be shared among the users of the network. This includes not only the possibility of sharing software, but also special hardware capabilities such as ILLIAC IV and also unique data sources such as large global weather data bases.

ARPANET was, and still is, a success. Following this lead several other networks have been implemented and many more are being planned. There is now COST-11 linking England, France, Switzerland and Italy; CYCLADES, a French network; EPSS, developed by the British Post Office; SITA, a special purpose network for European airlines and ALOHA, a network linking together countries in the Pacific area and Australasia.

1.2 The KIWINET project

In late 1974 and during 1975 Computer Centre and Computer Science Department staff from Victoria University, Wellington and Massey University,

Palmerston North, met occasionally to discuss the feasibility of setting up an experimental computer network linking the two sites. The aim of the network, which had been given the name KIWINET, was to enable the reduction of computing costs at both sites by sharing software and hardware resources.

Both project sites had identical computers (Burroughs B6700's) and the initial proposal was that the network would consist solely of these two machines. However it was felt that any future development of the network should not be restricted to Burroughs computers and that therefore any design of the initial network should make as few assumptions as possible about the type of computers comprising it.

In October 1975 finance was acquired for the rental of a Post Office telephone line between the two sites and shortly thereafter formal meetings were held to set up the research team and provide direction to the team's efforts. At these meetings it was felt that the project could be considered on four separate, but naturally inter-related, levels and four working groups, one for each level, were accordingly formed.

(i) Hardware Level

This level, conceptually the lowest level, deals with the data communication equipment on which the network will be implemented. The desired characteristics of this equipment (e.g. speed, reliability)

depend to some extent on the way the whole system will be used. Price/performance trade-offs must be made when deciding such things as line baud rates and what to use for front-end processors.

(ii) Data Communication Level

This level deals with the method by which messages are encoded and transmitted between nodes of the network. Decisions must be made concerning message protocols, and message switching software must be developed.

(iii) The Control Language Level

This level is concerned with the language(s) a user needs to control the network, commanding it to perform any desired tasks. Consideration must be given to what the user needs to do and a language to do it with must be provided.

It is this level with which this thesis is concerned.

(iv) Project Management Level

This, the highest level, is concerned with the services provided by the network and overall management of the project. Consideration is given to things such as the user charging system, resource co-ordination and project finance.

1.3 The Control Language Problem

The problem faced by the control language group is based on a problem faced by the users of all automated machinery since Frankenstein invented his monster; once you've got your program into a computer, once you've built your monster, you must have some way to control it, some way to tell it to start, perform certain tasks and then stop.

Nowadays, with computers, a big red 'GO' button is no longer enough. Modern-day computer users want to do complex things - they may want to fiddle with files copying them from one medium to another, renaming them and so on; they may want to execute one program only if another succeeded first; they may even want to run several programs simultaneously, synchronising their actions at several points.

The complexity of the requirements of modern-day users demands that there must be some language with which a user can control the computer, specifying what he wants done with his programs and files. Of course, it is not the computer that users control but rather the Operating System (OS) and, to be accurate, it's more a case of requesting the OS to perform the desired tasks, rather than controlling it.

There is a distinction between the language (or languages) used to program a computer and that used to control the operating system. The former are called Programming Languages (PL's) and the latter is called an Operating System Control Language (OSCL) or a Command Language (CL)*. Most modern OSCL's are designed for one and only one particular OS and no OS will understand the OSCL designed for another. To make the situation even worse each distinct computer series has its own, specially designed, idiosyncratic OS.

Now comes the problem. Most users who wish to do anything with a computer must use that computer's OSCL. (Admittedly, with some systems, e.g. the HP300, users don't have to do anything, being provided with Menus

---

* Further names are Job Control Language (JCL), Job Description Language (JDL) and just Control Language. We will use OSCL or CL only.

and Help facilities. These however are special cases and the more sophisticated users of these systems still require an OSCL.) Now, having to use an OSCL might not seem so bad for the user of a single machine but when that user is faced with a network of machines, each one with a possibly different OSCL, things get a bit difficult. Imagine Frankenstein trying to control an army of monsters each one of which speaks a different language.

1.4. The SCL decision

The control language team, confronted with the afore-mentioned problem had two choices. They could either have ignored it, providing simply a mechanism within the OS for identifying to it the site at which a particular job should be run and where files were located, or they could have provided a common control language for all machines in the network. After much discussion it was agreed that to ignore the problem was to ignore the needs of the network's eventual users and that a standard OSCL should be provided.

The team corresponded with Professor Sayani, Chairman of the CODASYL OSCL task group, and Professor Unger, Chairman of the IFIP Conference on Command Languages 1974. They replied that no standard OSCL had as yet been defined although there appeared to be a preference for a block-structured language. With this in mind a search for suitable block-

structured OSCL's was initiated. The team found only two:- ICL's System Control Language (SCL) developed for their 2900 series of computers, and Burroughs B6700/7700 Work Flow Language (WFL) with which the team was already familiar.

WFL is a batch language only. CANDE, a separate unstructured language, is provided for interactive use. These two languages are not orthogonal, different constructs being required to carry out similar functions. The team felt that a standard language should be usable with a minimum of differences in both batch and interactive modes. Futhermore it was considered that it would be difficult to adapt these languages to include the general handling of files and system resources and those features necessary to enable specification of where programs are to be run and files to be held.

At first sight SCL appeared to be more promising than WFL, offering a well constructed block system and powerful control structures, yet still remaining adaptable. Deeper investigation revealed that files and other system resources (e.g. peripherals) are handled by a large number of macros and procedures. The average user would need to know approximately 30 out of over 100 supplied and this was felt unacceptable.

However SCL was felt to be the better language and it was decided to use it as a basis around which a network control language could be designed.

At this point, in mid 1976, the author, then working for the Massey University Computer Centre, joined Mr Keith Hopper and Mr Neil James of the control language team and began work on the additions and alterations to SCL.

1.5 The aim of this thesis

The aim of this thesis is to present the author's work on the design of a standard OSCL for a network of computers.

The first stage of this work consists of defining the problem. The role of an OSCL is placed in relation to the complete interface between users and a network and the qualities and features that a standard OSCL should have are defined.

Having defined the problem the next steps is to determine what other people have done about it. A survey of work on the standardisation of OSCLs is carried out and several 'standard' OSCLs are evaluated.

The third stage of the work is to develop SCL, adding any features necessary for networks and making alterations to avoid the problems and disadvantages of previous OSCLs.

Lastly, although an implementation of the language has not been attempted, the problems involved in one are considered.