AN INVESTIGATION INTO THE RELEVANCE OF DOCUMENTATION FOR PROBLEM SOLVING IN EQUIPMENT CONTROLLED BY PROGRAMMABLE LOGIC CONTROLLERS (PLCS)

James Prendergast[1] and Michael Barrett[2]

[1] Institute of Technology, Tralee, Co.Kerry, Ireland
[2] FAS Training Centre, Athlone, Co.Westmeath, Ireland
Email: James.Prendergast@staff.ittralee.ie

ABSTRACT
An industry survey has identified a lack of support documentation for Programmable Controlled equipment in the manufacturing sector. To assess the relevance of documentation for problem solving on such equipment a series of experiments were carried out. The experiments tested the relevance of concise as against indifferent documentation for hardware and software maintenance. The results were stark– effective documentation leads to faster solutions to hardware and software issues.

INTRODUCTION
The Programmable Controller can arguably be said to be the common vital component that has lead to affordable automation across the manufacturing spectrum. The programmable controller is a very reliable industrial computer designed to survive in harsh environments.Research indicates that in the field of PLC programming the documented structured approach followed in computer software projects seems not to be generally applied [1]. Investigations have shown that the PLC programming is very “ad hoc”, documentation is poor and every PLC programmer has their own version of the correct way to write and implement programs. This “ad hoc” development approach is costing Irish companies millions of euros each year as the PLC programs used to control equipment and production lines are documented and written in a very inefficient fashion causing maintenance problems which in turn results in increased machine downtime.

DOCUMENTED STRUCTURED PROGRAMMING - THE ADVANTAGES
The advantages of documented structured programming have been verified by a research project carried out by Vinther Buus Industriraadgivning (ABI), a Danish company [8]. The objectives of the research which was to test whether it is possible and realistic to apply standardised and structured methods of documentation and programming to software projects (PC and PLC) found that standardised documentation has several positive results such as:

1. Facilitates more professional output;
2. Promotes reuse of code;
3. Increases throughput;
4. Facilitate maintenance

THE SOFTWARE DEVELOPMENT PROCESS
Traditionally the development of a software project followed the highly structured “waterfall” approach [3]. In the waterfall Software Development Life Cycle (SDLC)

model once a stage is completed it is never revisited. In practice however the waterfall model has evolved into an iterative system where information gleaned on one stage is used to modify a previous one as illustrated by Figure 1.

The Stages

1 Requirements Analysis

The purpose of this stage has two goals which are [4]:

1. To elicit a detailed description of what the proposed project must achieve;
2. To specify the external behaviour of a system to address those details.

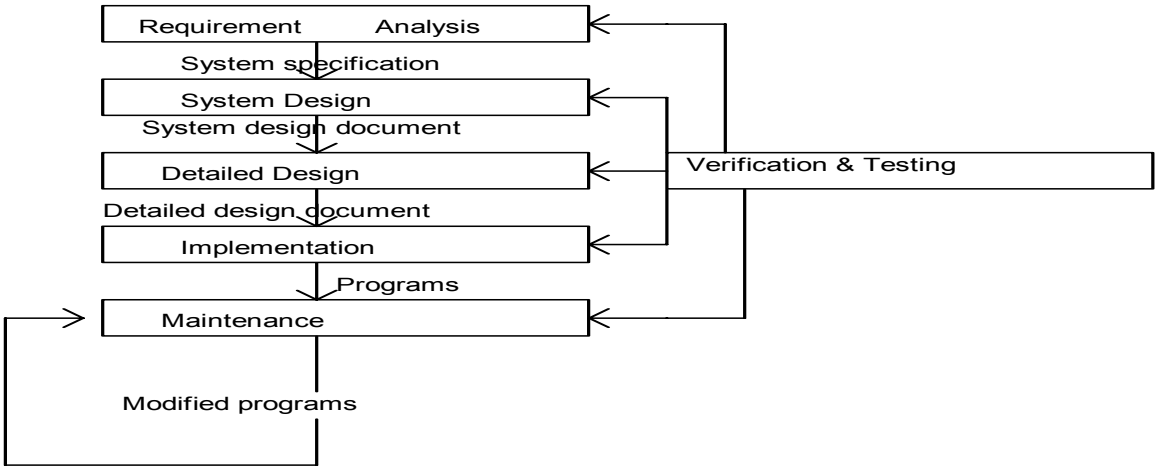

Figure1: The iterative waterfall process [2]

2 The Design Stage

The Design Stage is the intermediate stage between the natural language statements that describe the requirements the system must fulfil and the formal list of instructions that constitute a computer program. A number of techniques and program design tools are available for this purpose.

3 Implementation Stage

Writing the software using an appropriate programming language is the focus of this stage. The programming language can be Procedural such as Basic, FORTRAN, "C" or Object Oriented (OO) such as Visual Basic or Java. For PLCs the language will be IEC 1131-3 defined.

4 The Testing Stage

The purpose of testing includes finding defects, blocking premature release of the software, minimising technical support after release and conformance to standards [5]. A test plan should be drawn up to identify the best approach to testing, to itemise the features to be tested and to set pass / fail criteria [6].

5 The Maintenance Stage

Software Maintenance is the process of modifying software after it's completion. There are three categories [7]:

- Software update or Adaptive maintenance changes the functional specification;
- Perfective maintenance to enhance performance;
- Corrective or Adaptive maintenance, repairs a bug but does not modify any function.

EXPERIMENTATION

An industry survey identified poor SDLC documentation as a major issue in the majority of organisations surveyed [1]. Controlled experiments were carried out by the authors to attempt to quantify the effects of quality as against poor documentation in terms of their usefulness in hardware and software maintenance. The first experiment tests differing levels of SDLC documentation for hardware maintenance purposes. It was noted in carrying out the industry survey that maintenance personnel routinely used whatever documentation was available to find faults on automated equipment. For this reason experiment number 1 was carried out to ascertain the usefulness of such documentation in the hardware fault finding process. Software maintenance has been identified as a vital phase of the SDLC. The second experiment investigates the results of using poor and comprehensive documentation for software maintenance purposes.

Fault Finding

The Purpose of this experiment is to examine the hypothesis that a poorly documented PLC program makes hardware fault finding more difficult than a well documented one. The Subjects are a group of sixteen electrical Apprentices who have completed phase six of the Standards Based Apprenticeship. Because their training is common with the training that all electricians in Irish Industry receive they would be representative of that group of electricians involved in hardware maintenance functions. The Test Platform is a pneumatic test rig controlled by an Allen Bradley MicroLogix PLC with RsLogix500 software.

The subjects are divided into two Groups, “A” and “B”.
Group “A” are given a document containing:

- A description of the sequence of operation of the rig;
- A flow chart;
- The I/O Assignment list;
- A printout of the PLC program with detailed natural language comments and labels.

Group “B” are given a document containing:

- The I/O Assignment list ;
- A printout of the PLC program with sparse comments and labels.

All groups are given an electrical and mechanical drawing of the rig.
A hardware fault was introduced into the rig. Each participant is asked in turn to fix the fault.

Software Maintenance

The purpose of this experiment is to examine the hypothesis that a poorly documented PLC program makes Software Maintenance more difficult than a well documented one. The Test Platform consists of a Mitsubishi FX0 PLC set up to control two conveyors each being driven by a Three Phase Motor.

The Experiment Subjects

The Subject Groups (eight members per Group) are from the Industrial Automation Class in F.A.S. Regarding the representation criterion, the Subject Group has as a background trade or third level qualifications, they have completed the year long course in Industrial Automation and are thus representative of maintenance technicians at entry level in industry. They fall into Shneiderman’s scale for assessing

programming experience as Novices [9]. The experiment was run over three sessions of three different classes. The participants have just completed the PLC section of the course and are familiar with the Mitsubishi FX0 PLC and the GX Developer Software. The experiment required the participants to make three modifications to the program.
The participants are divided into two groups, “A” and “B”.
Documentation supplied to Group “A”:
Ladder detail: Good use of Statements, Comments and Notes;
Accompanying documentation:
1. Written description of process;
2. Flow Chart;
3. Program layout by Function;
4. I/O List

Documentation supplied to Group “B”:
Ladder detail: Well Commented but with no Statements and Notes.
Accompanying documentation:
1. I/O List

RESULTS AND DISCUSSION

Figure 2 indicates a clear relationship exists between the level of documentation supplied and the likelihood of a fault being found. By using the I/O List effectively all participants should have been able the find the fault, however, fault finding is not one of the objectives in the PLC module for phase 6 Apprentices. The conclusion that good documentation is a useful tool for hardware maintenance is valid, however, training in the effective use of such documentation will strengthen that usefulness.

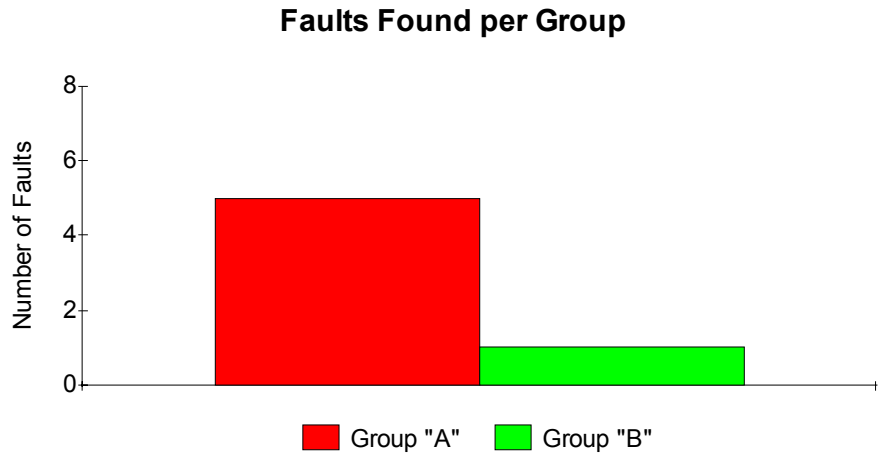

Figure 2: Faults found by groups

The program supplied to both groups is the same in terms of structure and Comments - the only difference being that the statements are removed from the program supplied to Group “B”. The number of corrected modifications to the program for each group member was added and graphed, see Figure 3.

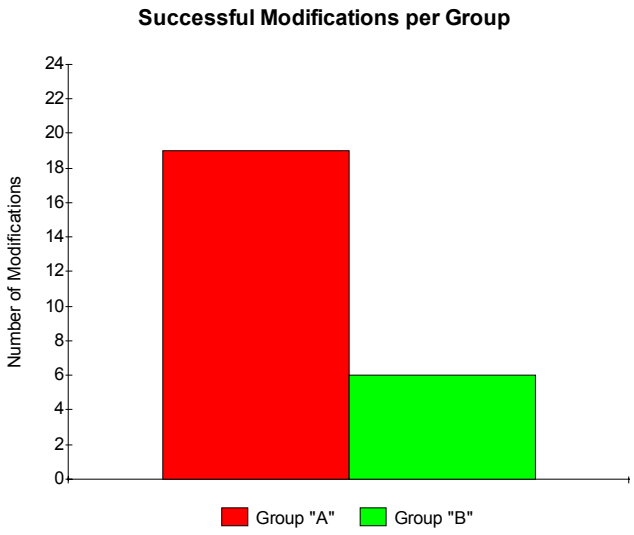

Figure 3: Number of correct program modifications by each group

As can be seen from Figure 3 members of the Group supplied with the full documentation were successful in completing nineteen out of twenty four possible (79%) modifications while group “B” members completed six out of a possible twenty four (25%) modifications. Brooks says that fixing a defect in a program has a 20-50% chance of introducing another defect [10]. It was therefore decided to examine each group members program to see if bugs had been introduced. As can be seen from Figure 4 a total of eight bugs were introduced by Group “B” as against four by group “A”. This indicates that attempting program modifications with inadequate documentation is twice as likely to result in the introduction of defects as when complete documentation is used.

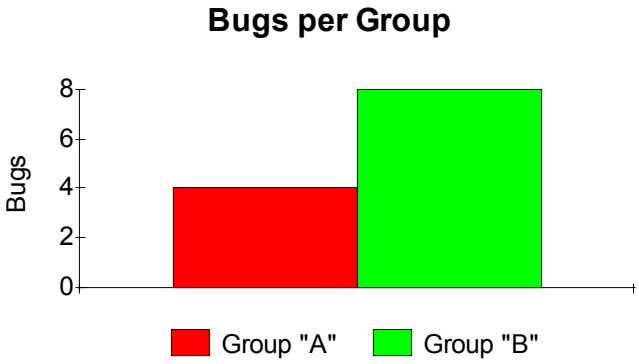

Figure 4: Bugs Introduced by each group

CONCLUSION

While the consequences of using inadequate documentation for problem solving in industry cannot be quantified any factor which makes the manufacturing sector less efficient must be viewed with concern bearing in mind the relocation of many companies in this sector to Eastern Europe and China. The experiments provide clear evidence of the advantages to be gained from using concise documentation for

problem solving. Training in the use of such documentation has the potential to facilitate more effective problem solving.

REFERENCES

1. Prendergast, J., Barrett, M., "Programmable Logic Controller Programming - A Proposed Protocol for Procurement and Development, with Special Relevance to Training and Education" Proceedings of the International Conference on Mechatronics, 2006, IEEE, Budapest, pp. 386 – 391
2. Woodlam, M., Heal, B., Introduction to VDM, The Open University, McGraw- Hill, England, 1993
3. Brookshear, Glenn, J., Computer Science an Overview, The Benjamin Cummins Publishing Company Inc., 1994, pp. 246 - 249
4. Page-Jones, M., The Practical Guide to Structured Systems Design, 1988, Prentice-Hall, New Jersey,
5. http://www.testingeducation.org/articles/what_is_a_good_test_case_star_2003_paper.pdf Kaner, C., What is a Good Test Case?, accessed 19/10/2006
6. IEEE Standard for Software Test Documentation, The Institute of Electrical and Electronics Engineers, 1998, New York
7. Boehm, B., Software Engineering Economics, 1981, Prentice -Hall, Inc. New Jersey
8. Vinther Buus Industriraadgivning A/S., Structured Documentation on small-scale Projects, 1996, Aalborg
9. Shneiderman, B., Software Psychology - Human Factors in Computer and Information Systems, 1980, Winthrop Publishers Inc, Massachusetts,
10. Brooks, Frederick P., The Mythical Man - Month, 1982, Addison-Wesley Publishing Company, pp. 122