SYSTEMS DEVELOPMENT

Basic flaws in the current culture
Ideas for rectifying some of the problems

M A KINGSBURY
formerly Principal Lecturer
Department of Computing
Staffordshire Polytechnic
England

Abstract

Systems Development: - Basic flaws in the current culture
- Ideas for rectifying some of the problems

It is now 21 years since the term 'Software Engineering' was coined. In this time relatively little progress seems to have been made in getting the development of computing systems under control. This paper identifies three basic flaws in computing systems development culture, it tries to identify the reasons for the flaws and suggests some ideas for rectifying them.

The flaws identified are:

(i) A lack of a quality culture in computing systems development.

(ii) Application domains not clearly defined/understood/researched.

(iii) Too often computing research and development seems to be fragmented and heading in all directions simultaneously.

An example of the lack of an understanding of quality and its consequence is as follows:

"The perception and assessment of the quality of things <u>with which we are familiar</u> is an accepted and natural skill, eg clothes, cars, accommodation, engine components, videos, literature, etc.

Also the people responsible for developing these products normally know how to achieve the required quality level.

Abstract (Contd)

With respect to computing systems, in general the doers and supervisors do not appear to have developed the skills of assessing the quality of a software product and/or its components. Hence how will then deficiency affect the selection of alternative sources of reusable code?

["It is not enough for everyone to do his best. Everyone is already doing his best." Dr W E Deming.]"

Much, probably most, research and development in computing systems is computing or software technology led (eg The Alvey Program).

The paper suggests that more research and development should be 'Application domain' led. Initially to identify the domains and the particular tools/skills/design needed, both technically and managerially.

If the 'Application domains' are understood then it is possible that software engineers will be able to move from being mechanics or technicians (equivalent to 1880 engineers) to engineers who understand the application area in which they work (equivalent to 1990 engineers). Also if research and development is application domain led then it is probable that the development of quality cultures in specific domain areas may accelerate.

Contents:

Section 1 - Introduction
Section 2 - Possible/Probable reasons for the failure of the Software Engineering Industry
Section 3 - Potential Areas of RAD
Section 4 - Conclusions
References

1. Introduction

It is now 21 years since the term 'Software Engineering' was coined. I believe that the Software Engineering task is to produce the

right product
on time
within budget.

In other words to produce software (or system) products of the correct quality level; using as a definition of quality -

fitness for purpose (right product on time)
and value for money (within budget)

Unfortunately, from personal observations and articles in the technical and national press, it would appear that the software development industry in losing the confidence of its customer base because, all too frequently, it cannot deliver goods of the correct quality level.

> "A recent survey showed that 28% of the clients of the UK's top accounting firms had suffered a computer disaster within the last 5 years
>
> Research from the US ..... shows that 90% of firms which suffer a major computer disaster have gone out of business within 18 months
>
> ..... in the hands of a bunch of roving mercenaries hired by a department with a track record of failing to deliver to time or cost ..."
>
> **THE TIMES**
> **3 May 1988**

The purpose of this paper is to:-

(a) Identify some of the reasons for the failure of the industry, these are probably familiar to most of us.

(b) To suggest some areas for research and development.

(c) The stimulate discussion.

2. Possible/Probable reasons for the failure of the Software Engineering Industry

Probably the prime causes for the problems facing the industry are:

2.1 A lack of a quality culture in the computing systems development industry.

2.2 Application Domains not defined/understood/researched.

2.3 Too often computing research and development seems to be fragmented and technicality led. ({He} flung himself upon his horse and rode madly off in all directions."

Stephen Leacock)

2.4 Education and training is primarily aimed at producing technicians, not software engineers.

2.5 Lack of education in design.

2.6 The expansion and growth of applications and application areas.

2.7 Lack of management skill and/or knowledge.

2.8 Staff turnover

"Some sections of the IT industry have to replace almost 25% of their staff every year because of the high turnover rate ....."

"..... it has been estimated that it could cost the IT industry up to half a billion pounds every year to replace staff."

THE TIMES
19 January 1989

2.9 Resistance to change both technically and managerially.

2.10 Little knowledge transfer from other areas/industries.

2.11 Poor record keeping.

3. Potential Areas for R & D

Hopefully the following areas for R & D cover 2.1 to 2.11, also it is probable that current R & D is already tackling some/most of the problems.

3.1 Lack of a Quality Culture.

("It is not enough for everyone to do his best. Everyone is already doing his best." Dr W E Deming)

3.1.1 This lack of a Quality Culture has probably been brought about in the UK because:

(a) The growth of the industry has been sustained by recruiting young people, frequently new graduates.

(b) Historically it appears to me that Universities have been primarily interested in research and that quality has been equated mainly with excellence. [Perhaps this is one of the reasons why we as a Nation think we are good at research but not so good at development ie turning research into marketable products.] To a large extent Polytechnics have imitated Universities.

Thus it is probable that undergraduates in computing will receive little or no education in quality; and will also gain little practical experience in it. [This is also probably true of the tutors].

Solution

Project A (i) - R & D activities within Universities/ Polytechnics which are funded by outside sources such as ESPRIT say, to be carried out under some defined QUALITY CONTROL SYSTEM (based on BS5750, MOD or NATO requirements say)

Project A (ii) - Computing departments within Universities/Polytechnics carry out their functions under a QUALITY CONTROL SYSTEM.

Experience gained from (i) and (ii) would naturally be fed back into undergraduate education, this should help to resolve 2.1, 2.4 and 2.9.

[Note. A QUALITY CONTROL SYSTEM defines what has to be achieved within an organisation, not how to achieve it.]

3.1.2 The perception and assessment of the quality of things with which we are familiar is an accepted and natural skill, ie we have a 'black box' assessment skill for clothes, cars, accommodation, engine components, films, videos, literature etc.

Also the people responsible for developing those products normally know how the quality is achieved (ie white box assessment).

With respect to software, again we probably have reasonable skill in assessing black box quality (as do our customers), it is unlikely that we are very skilful in 'white box' assessment of the product ie requirements definition, design, code, verification, validation.

Project B - More R & D into software metrics, hopefully supported from installations using IPSE's.

This could help to resolve 2.1 and 2.9.

3.2 Application Domains

Probably one of the largest areas of ignorance is understanding the boundaries of, or knowing the definition of, application domains; in any case they will not be absolute. Also it is probable that too much R & D is bottom up driven, that is development routes/tools/management control systems are designed for general purpose use in all (or most) application domains.

It is possible, that more cost effective software development systems capable of producing products to defined quality levels could be devised if we defined and understood the application domain, then designed the software development system together with the Quality Control system.

In a recent Cardiff Business School paper, "Manufacturing and Personnel Strategy in Western and Japanese owned Companies in Britain" by Nick Oliver and Barry Wilkinson the authors observe that:

> "Traditionally Japanese companies put their personnel strategies into practice at the same time as the new manufacturing and working methods."
>
> "..... the Japanese are introducing far more of the personnel practices for which they are renowned - highly selective recruitment, direct communication, long term employment for core workers ....."
>
> etc

This really confirms the need for total quality control, not just quality control of the technical activities.

Solution C (i) - Let us assume that Computing Departments in Universities and Polytechnics are an application domain area. One year after the start of solutions A (i) and A (ii) carry out a survey of the domain to confirm, or otherwise, that it has clear characteristics and a boundary; identify a 'best fit' quality control system; identify a 'best fit' software development route and personnel functions. (See fig 1).

Solution C (ii) - From experience gained from C (i) and/or in parallel with C (i) develop and produce domain definitions, domain specific quality control systems, software development routes and personnel management activities. (See fig 1).

Solutions C (i) and C (ii) may help to resolve 2.2, 2.4, 2.8, 2.9 and 2.10.

Application Domains (Contd)

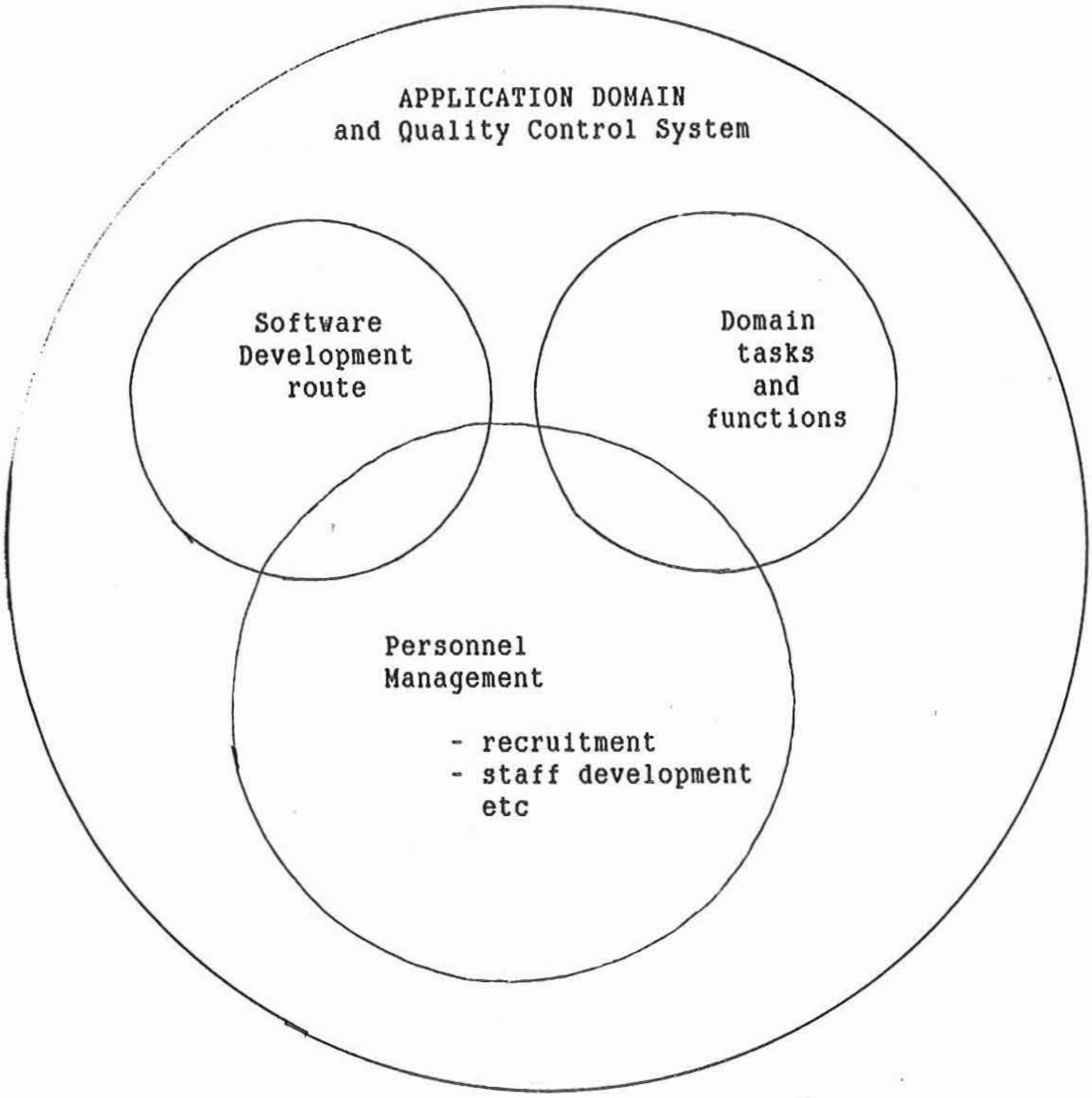

Fig 1

3.3 Other possible areas of R & D

3.3.1 Project D - Investigate how designs can be captured, compared and understood so that software engineers can gain from implemented successes and failures.
(Help to resolve 2.5)

3.3.2 Project E - Study the management and quality techniques of other industries which face or have faced similar problems to the software industry eg film and TV program production, civil engineering, building trade, architecture etc.

3.3.3 Project F - Study, review and compare the management control and quality methods and techniques used in software development in all cultures ie Western Europe
North America
USSR/Eastern Europe
Pacific Rim

3.3.4 Project G - Study and apply the developments in safety critical software.

4. Conclusion

It seems to me that currently software engineering is roughly in the position that traditional engineering had achieved in the 1880's, that is when engineers were mainly mechanics and technicians. For software engineering to move into the 1990's I believe that quality cultures, management methods and technical knowledge in specific domain areas needs to be developed.

If section 2 is substantially correct, then people who educate, train, employ and/or are software engineers should consider why this is happening, is it important, what can be done to rectify the situation.

If section 2 is substantially incorrect and the customer base is substantially satisfied then there is no case for the industry to answer.

References

THE TIMES - 17 March 1987

THE TIMES - 3 May 1988

THE TIMES - 19 January 1989

Manufacturing and Personnel Strategy in Western and Japanese owned Companies in Britain - by Nick Oliver and Barry Wilkinson, Cardiff Business School.