The state of the practice of methodologies in Australia

B. Wong

School of Computing Sciences, University of Technology, Sydney, P.O. Box 123, Broadway, NSW 2007, Australia

Abstract

Since the early seventies many different methodologies (commonly called development life cycles, process architectures) have been introduced to help develop quality software products. Many practitioners and researchers have the view that these methodologies lead to improved quality in both the development process and the software product. The research looks at the Australian industry as to whether this is the belief of the Australian practitioners. The research investigates a number of Australian organisations to see whether a methodology has been defined. Both the traditional functional decomposition paradigm and the object oriented (OO) paradigm are addressed. Methodologies have been identified as very important when it comes to improving the chances of obtaining a quality product. The results of the research showed that many organisations had a methodology for functional decomposition but very few (of those conducting OO development) had one for object oriented. The results also showed that in many cases where organisations had a methodology, the developers did not follow it or did not know of its existence. Because of the importance of developing a successful software product, higher education needs to focus on what it teaches. Both the methodologies practiced and not practiced need to be introduced and taught. Advantages and disadvantages need to be highlighted in the education of our students.

1. Introduction

The use of computers, both hardware and software, have become the central component in many of the activities of an organisation. From accounting, payroll, stock control to management needs, computers have been used to
improve efficiency and productivity. As these activities have grown in size and complexity there has been a need for more powerful and higher quality software. This has required continuous introduction of many specialised and powerful tools and techniques.

Unfortunately, the development of quality software has not been without criticisms and problems. Hirscheim & Newman[1] quoted the results of a survey conducted by Gladden in 1982. The authors stated that "75% of all systems developed, surveyed by Gladden, either the final product was never completed or the product of the completed development was not used, or the completed development did not work". The 75% value seems extremely high, in fact hard to believe. Whether we accept this value or not, the point being made here is that problems exist in software development. Other researchers have also commented on the poor quality of software being developed. According to DeMarco[2], the average software product on the market in the United States is not error-free. Boehm supports this by pointing out that more and more time is being devoted to maintenance in an average Data Processing division of a corporation (Boehm[3]). These increases of maintenance are due to enhancements and repairs. And Harrison states that software maintenance, whether enhancements or modifications or repairs, represents over half of the Data Processing division efforts (Harrison[4]). So how can we reduce this maintenance? Can we have a development process which increases the quality of software, resulting in reduced maintenance?

2. The Need for a "Methodology"

The introduction of development methodologies came about in the seventies as the way to increase software quality. From the Waterfall model (Royce[5]) to the Jackson Structured Programming methodology (Jackson[6]) to the Structured Analysis and Design methodology (Gane & Sarson[7]), different approaches were being created as the development process which works.

The term "Methodology" has been largely used in the commercial sector. Avison[8] described this term as "a collection of procedures, techniques, tools and documentation aids which help system developers in their efforts to implement a new information system". Jones[9] described this term as "a body of knowledge and techniques". Checkland[10] described it to be "a set of principles of methods, which in any particular situation has to be reduced to a method uniquely suited to that particular situation". The word "methodology" is not well defined in literature nor by practitioners. There is very little agreement as to what it means other than at a very general level. The term has been used very loosely and yet very extensively. Jones[9] says that there are many different "methodologies" in software engineering, each with its own partisans, and that individual techniques often can go by the name of "methodology". Infact, Longworth[11] in a study identified over 300. Bubenko[12] goes further and says that:
"it is a reasonable estimate that hundreds of more or less similar methodologies have been published. In practice, probably tens of thousands of more or less different approaches are being used. Most organisations have developed their own methodology and prescribed it in the organisation's handbook".

It may of course be, as Veryard[13] suggests, that the differences are often trivial and are made solely to differentiate these methodologies in the market place.

The many methodologies differ in tools, techniques, contents, differences in the philosophical view on which they are based (Avison[8]), from the phased approach to the systems approach to the humanistic approach. Whatever the differences, three main categories have been identified by Avison[8] as what people look for:

1. A Better End Product
2. A Better Development Process
3. A Standardised Process

Because of the poor definition given to methodology, we raise the following questions. What is the state of the practice in Australia for methodologies, what level of understanding do practitioners have on the topic of methodologies, and what can be done in higher education in relation to methodologies to help improve the quality of software developed.

3. The State of the Practice

We describe here a pilot study to review attitudes towards methodologies in both functional decomposition and OO in 30 Australian organizations. We wanted to understand how important methodologies were in Australian organizations, whether they had adopted a methodology, whether they are planning to adopt a methodology, what sort of methodology had been adopted, and whether they understand much about the methodology. We felt that an understanding of how methodologies had been accepted in industry would help us to appreciate the state of practice and help us to focus our higher education on improving the state of practice.

The organizations surveyed included a pharmaceutical manufacturer, public utilities, food manufacturers, a credit card company, finance companies, banks, an oil marketing company, insurance companies, software houses, government services, and computing companies. The companies in this study are a subset of organizations that currently employ, or recently employed students of the University of Technology, Sydney. The organizations range in size from about 25 employees, to several organizations having several thousand employees. The pilot study was conducted at the end of 1993. The data collected for this research involved surveys and interviews with IT management, systems
analysts, project managers, and programmers. In some organizations, more than one member was interviewed, sometimes from a different department. This gave a wider profile of the organization.

We were interested in whether methodologies were being applied for both OO development and functional development, what sort of methodology was used for both OO development and functional development, and what were the attitudes of the people surveyed towards the methodology.

4. Results

We found that most organisations professed to using a methodology for functional decomposition. However, very few companies had a formal OO methodology defined. The reasons for the lack of an OO methodology may be due to OO immaturity and a lack of appropriate automated support tools; or it may be because the OO approach is only adopted for part of the development (e.g. only OO Analysis etc.).

The results showed only 6% of organizations had a formal OO methodology in place. There were several reasons given for not introducing an OO methodology. They suggest that one of the major problems with OO was the lack of consistent and specific details regarding OO techniques. Official standards have not been set up by any groups, and there seemed to be confusion as to which direction (e.g. Booch) was the one to follow.

We were also interested in the types of methodologies being adopted. Most organisations followed a waterfall type methodology, though names had been changed. 2% were using Jackson Systems Development (JSD), 4% were using Structured Systems Analysis and Design Methodology (SSADM) and 20% were using Information Engineering. We asked about the awareness of
other methodologies and found that little was known about others. It seemed that prototyping was the most well known whilst methodologies like social technical, soft systems were never heard of. There is a need to improve the awareness of methodologies. How can an organisation choose or design a methodology effectively if one doesn't understand the topic fully?

![Diagram of Methodologies](image)

Figure 2: Types of Methodologies used in Functional Decomposition Development

Staff members were also questioned about their views towards the success or failure of the methodology. A number of staff members from different organisations stated that short cuts were often taken to get around the system. Junior staff or trainees were often not aware of the methodology, nor were they trained in it. They had a lack of understanding and therefore a lack of appreciation for the processes defined. In a number of organisations managers said their methodology training was given only to the management staff. This explains the ignorance displayed by the junior staff. It was also found that in a number of organisations the management staff identified a methodology in place, whilst the more junior staff were totally unaware of the existence of the methodology. The management were not even aware that the junior staff were not following the methodology. Attitudes towards having a methodology was found to be mixed. In some organisations management praised the methodology, whilst the junior staff were critical and negative towards it. In one organisation, a senior manager stated his scepticism towards the organisational methodology. His view was "my way has always worked, why should we follow this garbage?" Most people surveyed who had a formal methodology accepted it as a non evolving, non changing development process. This view demonstrated how incorrect some of the perceptions were towards a methodology. And finally, the attitude of a manager who had created their company methodology believed their methodology was perfect. In contrast we were fortunate to have developers from this company. They criticised the
methodology, highlighting how it had failed. They also stated that the manager and the other creators couldn't accept the criticisms, they couldn't believe their baby needed to evolve and improve. How can higher education help to reduce such problems?

5. Applying the State of Practice to Higher Education

Our research had indicated that many companies used a methodology of some sort for traditional development. However, though there was a methodology in place, it did not mean that it was successful. Nor did it mean that all staff understood, practiced or believed in it. We felt that much of the problems seen in this research stemmed from a lack of education and training. Software developers needed better understanding on the topic of methodologies. They needed to be shown the advantages and disadvantages, not just about the waterfall model but also the many other models. Developers and potential methodology designers needed to gain experience in creating a methodology. They needed to be able to improve their designs, and be able to accept and give positive criticisms.

The results of this research influenced us in the development and running of a post graduate subject "Quality and Software Engineering". The subject was part of a post graduate course in Software Quality Assurance. It focused on the tools and techniques available to software development and looked at how they could help in producing quality software. The topics covered in this subject were Development Methodologies, Project Management, Communication Theories in Information System, and Human Computer Interaction. Each topic area had often been thought of as an individual solution to software development, and so it was in this subject that we tried to highlight the importance of using all the techniques, emphasizing how the individual areas should complement each other.

Because of the results obtained from the survey, we modified the section dealing with methodologies. We believed that it was important for our graduates embarking in careers in Software Quality Assurance to be able to understand, build, modify, and improve methodologies. We believed that the many tools and techniques available to an organisation for software development be appropriately applied. We also wanted the students to understand the difficulties and problems they would face when creating a methodology. It was our hope that through better education, the negative views towards methodologies would be turned into positive views.

Because of this, students in this subject experienced a study on methodologies. They not only studied the advantages and disadvantages, but were also given the task of creating their own methodology, taking into account these advantages and disadvantages. The students then presented their creation, whilst their peers critiqued it, highlighting weak points and points of failure in the design. The aim of this exercise was to put the students through the experience of what the creator and what the skeptical developer would feel
whilst at the same time demonstrated the importance of continual evaluation and improvement of the methodology. The students learnt very quickly that through the eyes of the designer the methodology may have seemed complete, however from the perspective of the software developers it contained weaknesses which needed to be dealt with. A number of students were extremely confident that they had created the perfect methodology. It was interesting to see the change in their faces as their fellow students highlighted the weaknesses in their design.

We were very happy with the feedback from the students at the end of the semester. Many commented on how the subject had changed their view towards methodologies - from being ignorant and closed minded to now being open minded and appreciative. Others commented on how the section on methodologies helped them realize how difficult it was to design one and that designing a methodology should be an ongoing process.

6. Conclusion

The subject "Quality and Software Engineering" aims to develop the areas found weak in industry. If one is to improve the state of the practice whether in the topic of methodologies or any other area, higher education needs to be involved. They need to work together to produce better practitioners. Evaluation of industry needs to be performed and the results analysed so as to give recommendations to make higher education more appropriate to the real world. It is only then that future practitioners will be able to cope and deal with the current problems of the computer industry.

Acknowledgments

The author would like to thank the students from the School of Computing Science, University of Technology, Sydney, who provided some of the data used in this report.

References

10 Software Engineering in Higher Education