

https://dspace.org/

Pitfalls and guide lines in the transition to object oriented software design methodologies

Jansen van Rensburg, Miranda

1998
https://hdl.handle.net/10539/25463

Downloaded from a DSpace repository. https://dspace.org/

Pitfalls and guidelines in the transition to object oriented software design methodologies

Miranda Jansen van Rensburg

A research report submitted to the Faculty of Engineering, University of the Witwatersrand, Johannesburg, in partial fulfilment of the requirements for the degree of Master of Science in Engineering.

Johannesburg, 1998

DECLARATION

I declare that this research report is my own work. It is being submitted for the degree of Master of Science in Engineering in the University of the Witwatersrand, Johannesburg. It has not been submitted before any degree or examination in any other University.

(Signature of candidate)

3rd day of November 1998

ABSTRACT

Due to the dynamic nature of the software engineering industry there is a constant move towards new strategies for solving design problems. More specifically there is a move towards Object Oriented (OO) methodologies, presumably because of the various advantages offered in terms of maintainability, and reuse of code produced this way. As with various other aspects of the software industry there are however also problems encountered in this transition and lessons to be learned from the experience of companies who have already performed this change.

This research report investigates possible guidelines for companies who are currently contemplating a change to the OO software design methodologies, by covering a collection of issues one should know about prior to this change. It also summarises the problems faced in the transition so far, the reasons for these problems and suggests possible solutions. Lastly it also investigates new trends in the OO arena. The emphasis is on South African companies and projects. The results obtained are compared with results obtained overseas to find out what the differences and similarities are. Areas of concern are also identified, where theoreticians' views have been ignored, and both South African and overseas companies have not implemented any of the suggestions made.

ACKNOWLEDGEMENTS

I would hereby like to express my appreciation towards the following people and organisations for their contributions towards my research:

- The Foundation for Research and Development (FRD) for the funding they have made available towards this research.
- Technikon SA for their contribution in the funding of my research.
- My supervisor, Prof Barry Dwolatzky for his interest and guidance.
- All the representatives in the various companies both in South Africa and in the United States who have participated in interviews.
- The assistance of the Technikon SA library staff.

TABLE OF CONTENTS

Page Number/
Document Number

Declaration ii

Abstract iii

Acknowledgements iv

Table of Contents v

Foreword vii

Project Overview NDM110

MSc Paper 1 – M Jansen van Rensburg NDM120-20

MSc Paper 2 – M Jansen van Rensburg NDM120-21

Discussion and Conclusions NDM130

Bibliography \ References NDM150

Appendix Group I: Management Products

Master Document List NDM001

Project Management Plan NDM005

Appendix Group II: Technical Products – Literature Survey

Literature Survey NDM131

Appendix Group III: Other Technical Products

The Research process NDM309

Conclusions from informal interviews NDM305

List of companies NDM301

Questionnaire for telephonic interviews NDM302

Results from telephonic interviews NDM306

Conclusions from telephonic interviews NDM307

Local Interview Questionnaire NDM304

Conclusions from local interviews NDM310

Overseas Questionnaire NDM311

Results from overseas interviews NDM312

FOREWORD

This MSc Research Report was completed under the auspices of the SEAL, (Software Engineering Applications Laboratory) a SABS ISO 9001 listed enterprise. This provided an infrastructure for the development of the project, with all products developed, being audited at regular intervals.

The format for the report therefore differs from the conventional format in that it comprises a short body (in the form of papers) and a number of appendices. It is therefore considered helpful to provide the reader with guidance regarding the order in which the various documents should be reviewed.

The body of the research report will now be discussed:

The first document, numbered NDM110, contains an overview of the scope of this project.

For the essence of the project, the reader is directed to the following papers entitled

> *"Progressing towards Object Orientation in South Africa" (NDM12020)*

and

> *"Pitfalls and Guidelines in the Transition to Object Orientation - A global view" (NDM12021)*

The substance of the project will be found in these papers, whereas the appendices should be regarded as additional sources of information in understanding the issues at hand.

The document numbered NDM130 contains important information regarding the conclusions that were reached.

Lastly, the collection of references used for this project can be found in the document numbered NDM150.

The appendices within the research report are discussed below.

Two SEAL management products are included:

> The Master Document List (MDL), document number NDM 001, is a register of all documents created within this project.

The Project Management Plan, document number NDM 005, provides an overview of the required resources to produce the research report. It describes how the project was managed, by including a work breakdown structure.

The Literature Survey, document number NDM 131, investigates the literature currently available on the transition to OO (Object Orientation). It provides an overview of the issues involved in the adoption of OO. It consists of a summary of the technology itself, the benefits associated and then goes on to describe the situation by looking at the human resource issues, then the corporate issues and lastly the technical issues. Finally it also investigates what the future holds for OO.

Several project specific technical products were developed during the course of the project and are also included:

Firstly, the research process that was followed is described in the document numbered NDM 309.

The conclusions reached during the informal interviews are given in the document numbered NDM 305.

The list of companies used during telephonic interviews is documented in the document numbered NDM 301. The questionnaire used for these interviews is contained in the document numbered NDM 302. The results obtained and conclusions reached from these interviews are given in NDM 306 and NDM307 respectively.

The document numbered NDM 304 contains the questionnaire used during the more detailed local interviews. The conclusions reached from these interviews are given in the document numbered NDM 310.

Lastly, the questionnaire and results from the overseas interviews are contained in the documents numbered NDM 311 and NDM 312 respectively.

2 Project Overview

Due to the dynamic nature of the software engineering industry there is a constant move to new strategies for solving design problems. More specifically there is a move towards the object oriented (OO) methodologies, presumably because of the various advantages offered in terms of maintainability and reuse of code produced this way[7] . As with various other aspects of the software industry there are however also problems encountered in this transition and lessons to be learned from the experience of companies who have already performed this change.

There is therefore a need to know what these problems are and how to avoid them in order to make a success of OO projects, so that the advantages offered by OO can indeed be utilised.

The emphasis throughout this research project was on South African companies and projects. The results obtained locally were however also compared with the situation in the USA, as well as the situation documented in the literature available.

The research provided guidelines for companies who are currently contemplating a change to the OO methodologies, covering important issues one should know about prior to this change.

It also summarised the problems faced in the transition so far, the reasons for these problems and suggested possible solutions. Lastly it investigated new trends in the OO arena.

The research process that was followed will now be described.

Firstly, a literature survey was done to determine what is already known about the transition to OO (Object Orientation). This provided an overview of the issues involved in the adoption of OO.

A number of informal interviews were also conducted with selected South African companies to again determine what the important issues involved in a transition to Object Orientation are.

Once these issues were known, short telephonic interviews were held with a large number of South African companies in order to find a selection of companies that could be used for the final interview. Companies were chosen for the telephonic interview in a manner that represented all types of companies so that results would not be biased. This interview also served to gather some statistics regarding progress in the OO arena in

South Africa. For this purpose, the format of the questionnaire was structured using multiple choice and simple yes-no answers.

Next, more in depth interviews were held with a smaller number of South African companies chosen from the original list. This interview delved deeper into the issues that had arisen from the first informal interview, the literature study as well as the telephonic interviews. It described the progress in South African companies towards OO, while at the same time verifying some of the trends that had been discovered during the telephonic interviews. Finally it extracted the opinions of the respondents regarding future trends for OO. In contrast to the telephonic interviews, questions for this interview were formulated as issues for discussion rather than simple yes-no answers.

Lastly, once it became clear what the situation regarding OO in South Africa was, these results were compared with results from a selection of companies in the USA, to create a clear view of the differences and similarities locally and abroad. Since a large amount of literature was already available regarding the situation in the USA, a smaller number of companies was chosen for these interviews. Again, questions were formulated as issues for discussion.

As the research progressed, a distinct collection of issues was found to be relevant. These issues were the following:

- Companies' reasons for choosing OO
- The time when the adoption of OO takes place
- Training of OO
- The role of management in the transition
- Quality processes
- CASE and metrics
- Reuse
- How to handle legacy systems
- Factors influencing the speed of the adoption of OO
- Organisational structure
- Profile of the first OO project in the company
- Profile of the company, regarding size, market sector, etc.

- The usage of various OO methodologies
- Programming languages
- Testing of OO systems
- Future developments for OO

The progress towards the adoption of OO in South African companies was measured in terms of these issues. This progress is described in the first technical paper.

The second technical paper investigates these issues again, this time concentrating on the comparison between the situation in South Africa, the situation described in available literature and the situation in the USA. The comparison was again done in terms of these issues.

The conclusions that were reached can be used to guide more South African companies in the adoption of object oriented software design methodologies. Some of the lessons learnt will also apply to the future changes in software design methodologies.

■

Progressing towards Object Orientation in South Africa

M Jansen van Rensburg

Software Engineering Applications Laboratory, Electrical Engineering,
University of the Witwatersrand, Johannesburg, South Africa

Summary

This article provides a description of the current state of Object Orientation in South African companies. It forms part of a MSc research report, looking at the pitfalls and guidelines in the transition to OO. The problems involved in this transition are discussed and possible solutions and guidelines are provided. The future trends for OO are also investigated. The findings are presented as a collection of the issues involved in this transition.

Keywords

Object Orientation
Methodology

Computing Review Categories

Software Engineering

1. Introduction

Due to the dynamic nature of the software industry there is a constant pressure to adopt new design and implementation strategies. In particular, the past few years have been characterised by a move towards Object Oriented (OO) methodologies. This move has been motivated by a number of advantages associated with the OO approach such as improved maintainability and reuse of code. The transition to OO, however, has not been easy and lessons can be learned from the experience of companies that have already attempted this change.

There is a need therefore to understand the problems associated with adopting OO methodologies within an organisation, and to explore ways of avoiding them. This paper focuses on determining the progress that companies in general, and those in South Africa in particular, have made so far in the transition to OO.

2. The process

This paper reports on a research project which was structured as follows:

Firstly, a literature survey was conducted to determine the current state of knowledge in explaining and analysing the transition to OO methodologies. Simultaneously, a number of informal interviews were conducted with selected South African companies to explore their views and experiences. Based on the literature survey and these informal interviews a questionnaire was developed and administered telephonically to a selected sample of 120 South African companies. This telephonic questionnaire served to highlight important trends and issues in the transition to OO techniques.

Based on the responses to the telephone interviews a group of 12 companies was chosen from the original sample of 120. Each of these were visited and a face-to-face interview was conducted with the IT or software development manager. This interview delved more deeply into the issues that arose from the first informal interview, the literature study and the telephonic interviews.

This research process served to describe the progress towards OO in South African companies and to construct an empirical basis for an analysis of this transition locally and a comparison with international experience.

3. The Telephonic Interview

Telephonic interviews were conducted during May 1998. Questions were formulated with simple Yes / No or multiple choice answers.

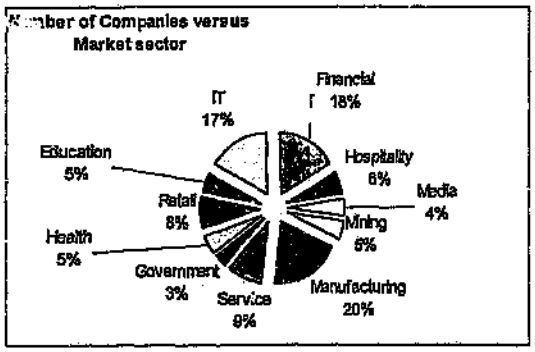


Figure 1Companies versus market sector

Since the results obtained from the telephonic interview would critically influence the remainder of the research, a representative sample of companies had to be obtained. Many companies operate nationally with offices in various regions. For this reason geographic location was not used as a basis for drawing the sample. Instead, market sector proved to be a useful criterion. Eleven market sectors were identified and a sample of 120 companies was selected in proportion to the size of the relevant market sector. The list of companies was compiled using several business directories and Internet classification directories as guidelines. Figure 1 illustrates the sample used in the telephonic interviews in relation to each company's market sector. Although geographical position was not used in drawing the sample, Figure 2 illustrates the composition of the sample in relation to the location of each company.

4. The detailed interviews

The detailed face-to-face interviews were conducted during June and July 1998. The decision to select companies to be interviewed was based on the extent of their OO experience and the importance of their market sectors. A set of questions were prepared in advance and were used as issues for discussion rather than simple Yes / No or multiple choice answers. The experience with OO in these companies ranged from "experimental" (less than one year) to eight years.

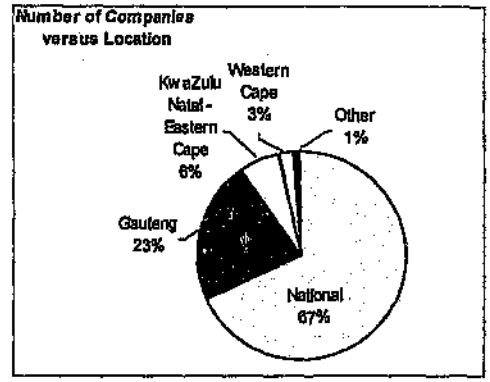


Figure 2Companies versus location

5. Results of the research

The remainder of this paper is structured around various issues which were identified in the published international literature. In each of the following subsections the issue is described and related to the situation in South Africa as determined from the research.

5.1 Why OO?

There is considerable agreement worldwide that choosing OO as the basis for software development offers advantages, including:

- Productivity through reuse: The 1997 Cutter Consortium's report[12], for which more than 200 enterprises world-wide were used in a survey, concluded that companies are adopting OO to achieve increased productivity through reuse;
- Higher-quality systems[23]
- Higher quality development process [23];
- Capacity to build larger systems[7] : "For large problems where a large staff is needed OO leads on each facet of the required teamwork. Once the initial definition of classes is started, teams can start development in separate groups of classes with less ripple effect between classes when changes are needed. The granularity of development tasks leads to a

natural strong fire wall between OT and most other classes." [7]

- Solution to a business problem: to promote OO with management, it is necessary to present OO as a solution to a business problem, says Litvintchouk.[17]

- Rapid development: The semantic richness of object models makes specifications more reversible and supports rapid application development directly. Sound object oriented analysis helps to deliver the benefits of OO much earlier in the life cycle. [19]

- Encapsulation[19]

- Extensibility and flexibility: Moreau in his article mentions the late binding of messages to target objects which provides flexibility to reconfigure systems dynamically without recompilation. [19]

In both the telephone questionnaire and the detailed interviews, the above advantages were explored in relation to the reasons for adopting OO in South Africa. The research showed that South African companies gave most of the above reasons for choosing OO. In particular they listed the following advantages:

- handling complexity;

- the promise of portability;

- maintainability;

- understanding the business;

- a need to model data and functions together;

- long term benefits and reuse. (Surprisingly, reuse was very low on the list of most companies. Therefore this issue was explored further and is discussed later in this paper.)

The research also showed that South African companies listed reasons for adopting OO which were not mentioned in the international literature. These include:

- improving quality

- "All the new technologies are in OO" - there is a need to move with the new technologies available

- Developers' needs – developers do not want to work on mainframe systems. Companies want to keep the right people and therefore are forced to move to the new technologies.

The last two reasons mentioned are issues of concern, as Page-Jones classified both as being the wrong reasons for adopting OO.[23]

5.2 Resisting the change

The detailed interviews revealed that before adopting OO most companies were using structured methods and that COBOL was the most widely used language. The next logical question was therefore to explore the issue of resistance to change towards OO.

Litvintchouk's research showed that organisations often undertake OO technology projects without the necessary management support. People adopt resistive attitudes where there isn't an atmosphere fostered to make it happen, which causes failure.[17] Vayda[32] found that, specifically in large companies, there was a great deal of inertia and resistance to change.

Our research shows that in South Africa, in those companies where OO was implemented with a clear understanding that it represented a paradigm shift and that design and implementation methods were to be changed (i.e. full OO), the process was met with considerable resistance. The resistance seemed to stem from the following:

- It seemed to be the older developers who did not approve the move to OO.

- Often being uninformed, developers, users and managers thought of OO as vapourware (a lot of hype but no results).

- At the time of the transition, developers did not understand the technology and thought it was too complex. Developers typically did not see the need to change

The companies which report very little resistance to the transition to OO appear to be those which either employ people with some prior knowledge

or experience of the technology, or people who are largely ignorant of what the new process entails. This also relates to the observation that people are only using the tools, and not the methodology. This reinforces Orr's [25] observation in 1995, that it is not clear how many people who think they are doing OO programming, actually do OOP, and that it seems to be a small percentage.

In his article Wick[33] provides a description of the content of a first year computer science (CS1) course and also describes how it should be adapted so that students learn that OO is purely a medium and not the message. The message should be software reusability. "The major pitfalls of current approaches to teaching C++ and OO in CS1 stem from a misplaced focus on the tools rather than on the application of the tools"..."Students are not motivated through the use of concepts before they are asked to consider their implementation". He also reasons that life cycle issues such as maintenance and documentation should be made important, arguing that CS1 courses often only teach students to be consumers not producers of reusable software.

5.3 Languages

The January 1998 survey in Object Magazine[22], found the primary programming language among respondents to be C++ (49.3% of respondents) while other languages included Java (18.7%), Smalltalk (16%) and Visual Basic (4%). C++ and Java were also the preferred languages according to the Cutter consortium report[12]. Reed's[26] explanation is that "because C++ is considered the successor to C, C programmers wanting the benefits of the object oriented paradigm look to C++ as a logical step toward object oriented programming".

In South Africa, C++ dominated the market, with Visual Basic and Borland's Delphi competing for being the second most used. Java and Smalltalk shared third place. Java is chosen for platform independence and pure OO, while Smalltalk is a popular choice when looking for a pure and dynamic language that also offers garbage collection. The research agrees with Reed's explanation in that South African programmers also see C++ as a natural progression from using C. It would appear from the research that companies in South Africa using Java, Smalltalk and C++ had more experience in OO, while companies using, for example, Visual Basic were using it for its attractive tools, rather than its OO features. A recommendation from a consulting company is that the implementation language should be chosen only after the architecture and a controlled development process has been put into place.

5.4 Testing

From Arnold's article[1] it is clear that testing is often the last item on the agenda and is therefore most often neglected: *"In the real world of large projects involving legacy systems, non-OO interfaces, non-infinite resources, non-infinitely applicable tools, competing and clashing features, and non-infinite windows, testing is thrown back into the trade-off world along with all the other trade-offs involved in engineering and business."*

From the interviews conducted as part of the research it appeared as if testing does not receive much attention in South Africa either. No mention was made of automated testing. The lack of a testing process in OO projects seems to be due to a lack of tools (for Smalltalk and Java), the high costs associated with tools, or tools covering only limited sections of the system life cycle. A further reason given was a lack of user commitment, since in many of the cases the testing also needs to include acceptance testing by the user.

The testing methods that were being used included "rigorous" testing in parallel with the old system running, using manual regression, code based integrated testing, and using information models as guidelines to define certain test areas. In many cases it is the developers who still do the testing themselves, and companies having separate testers seemed to be the exception rather than the rule.

5.5 Legacy systems - an unwanted reality

The significance of working with legacy systems can be found in Baer's comment: [8]

"The ability to identify successful strategies for working with [legacy systems] will determine the pace at which large organisations can move to object technology."

Vayda[32] recommends four strategies for reengineering legacy systems, including incremental reengineering, database conversion[1], migration strategies (such as running parallel systems until the new system is in place) and wrappers.

In South Africa it seems from the research that most companies have systems which they have classified as legacy systems. Interestingly, in a number of cases these systems were relatively new (3 years), and some were even OO systems.[2] Methods of dealing with these legacy systems included:

- mapping the problem with a CASE tool,
- using messaging and CORBA for wrapping,
- using queues to integrate where there is data entry in multiple places,
- file integration and data conversion (where data is the only contact between the old and new systems).

5.6 The promise of reuse

"The sophistication a company shows in its use of classes and components is a good sign of the company's overall progress toward an OO based development capability" – Harmon [12]

There is currently little empirical information on what to expect from reuse in terms of productivity and quality gains.[3] Vayda[32] distinguishes between organisational barriers and technical barriers to achieving reuse.

Organisational barriers include the transformation of the organisational mindset to include reuse, developing resources to champion reuse, and developing reward structures, with reuse specialists being rewarded quite differently from developers and developers being rewarded for reuse. Key personnel have to be taken away from other tasks in order to bring sufficient business knowledge to new OO developments.[11]

Technical barriers include the difficulty in producing truly general reusable components, documenting and distributing the components, finding the right components, handling changes to the libraries and namespace conflicts when integrating libraries from multiple vendors. One should also include over-generalisation. [11] Korson[14] warns against so-called "Libraries of sofabeds" which are classes that have a large reuse potential but are not optimal for any given application. Finally, the inclusion of application specific ideas in a library intended for general use, leading to restrictions and delay in subsequent projects,[11] as well as the unavailability of tools for supporting reuse,[15] were also given as reasons for not achieving reuse successfully.

Being a difficult process, reuse was not a primary objective for many companies in South Africa. Reuse is often tied to the type of business, and development is therefore often too specific rather than too generic.[4] In the case of a company from the military sector, which was interviewed, it was pointed out that in their applications the software is embedded and the hardware is changing. They therefore believe that it is impractical to try to achieve reuse. In the interview with a consulting company they proposed promoting maintainability and risk management rather than reuse. Other suggestions coming out of the local research were using refactoring[3] after development, writing a framework and making that specific, and concentrating on having the same development process throughout. According to a large insurance company, reuse comes from repetition by going through multiple abstractions. Trying to design for reuse was considered to be premature since it

---

[1] Controversy exists on this issue. According to Korncoff[8] legacy databases should be left intact.

[2] Casais also referred to this problem in his article[9]

[3] Often when changing code, additions become very complex but there is no time for redesign. Refactoring describes the techniques that reduce the pain of redesigning. The functionality of the software is not changed, but rather the internal structure. It normally involves small steps at a time, such as moving a field to another class. Also, while using refactoring no new functionality is added. The technique is still new and is mainly used in the Smalltalk community but promises to improve software development in all environments.[6]

cannot appear up front in the design. The message is therefore to concentrate on use rather than reuse. Some of these companies are however investigating appointing a person as a "reuse miner" to manage the libraries, plan the repository, etc.

5.7 Where South Africa lags behind – organisational structure

From opinions expressed in the international literature, it is clear that organisational issues have a large part to play in the successful implementation of OO in a company. In Graham's opinion, [10][11] "the real nightmare is the organisational issue. It has to come from the top." The focus here is on what happens to the project team structure, management policies and development process, says Korson[14]. He also claims to have seen "projects fail to achieve the promise of OO because the organisation did not understand the changes in corporate infrastructure that were necessary to support the object paradigm".

In 50% of the cases studied in the research, the organisational structure has not changed at all. Most respondents thought however that such a change was necessary and would happen in the future. The perception is that there are different roles for team members involved in OO development as compared to procedural development. Examples of such roles are a system architect who can act as mentor, a project manager, developers and a quality assurance team. In the cases where the company's organisational structure has changed, it seemed to be related to general restructuring and not related to OO. It is therefore clear that the organisational change related to OO is an area South African companies need to address urgently.

6. Experiences

In the 1997 Cutter Consortium's report[12], 41% of companies reported at least one OO failure. Reasons given included poor management and a shortage of experienced developers.

Exploring this further during the local interviews, one of the questions probed companies on the mistakes that they have made so far, together with problems they were currently experiencing. Some of the issues listed below may not seem directly OO related, but have led to a delay in the advance to OO in South Africa.

- Lack of proper design, and not using a methodology. The importance of the methodology was often not stressed sufficiently.
- Lack of skills, coupled with having no time for training.
- In the case of a large merchant bank, using a relational database – in this case 30% of the code and 60% of the performance were compromised just to handle the conversion from a relational to an OO database.
- Not having the luxury of experience gained on a smaller project.
- Not having the luxury to investigate different tools first.
- Many companies are still operating at a SEI CMM[4] level of one, thereby having "hero programmers" – if these programmers leave there is no one to take over.
- Lack of proper documentation.
- Lack of full time project management.
- Being too calendar driven – deadlines have to be met regardless of the quality of the system being developed.
- Lack of communication amongst users and between users and developers.
- The paradigm shift, requiring a new way of thinking.
- At a large defence industry company it was found that despite using OO for handling complexity, the system analysis was still functional in nature. It was difficult to find a

[4] The CMM (capability maturity model) was developed by the SEI (Software Engineering Institute) at Carnegie-Mellon University, and defines five levels of maturity in the software process of organisations.[19]

transition from analysis to design. There was no automatic process for integration of the whole life cycle.

Some of these issues are discussed in more detail in the following subsections.

6.1 The state of training

The Cutter Consortium Report[12] concluded that companies acquire the OO developers they need from the following sources: 51% train current staff in OO, 41% recruit staff with OO skills, 8% use consultants. The January OMO survey[22] found that 42.1% of respondents had some form of classroom training in their primary methodology, whereas 57.1% had not.

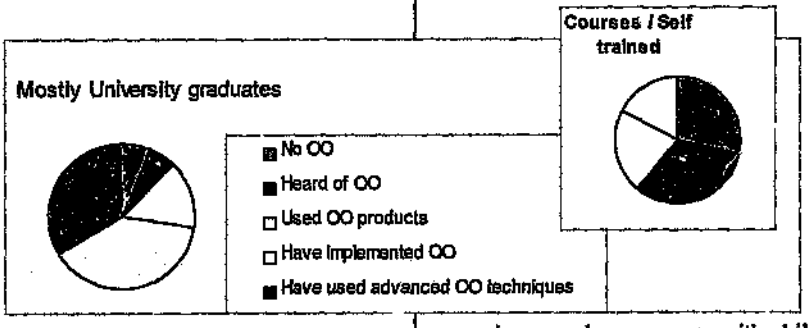


Figure 3Training and OO

Figure 3 illustrates the direct relationship found in the telephonic survey between skill level in South African companies and the experience in OO. The pie chart on the left indicates a large percentage of companies using mostly university graduates, lying in the "Have implemented OO" and "Have used advanced OO techniques" categories. The chart on the right (companies using self-trained staff) reveals a large percentage of companies in the "No OO" and "Heard of OO" categories. The question that therefore arose was: Is formal academic education a prerequisite for the successful implementation of OO? The detailed interviews revealed the following:

6.1.1 General requirements:

Companies generally were not specifically attempting to employ graduates. The policy seemed to be rather one of recruiting IT staff with a knowledge of the company's area of business or people with skills in the business who also had IT skills. People with practical experience are in demand. A close match between employees' values and the company culture will ultimately determine whether they will remain with the company in the long run.

6.1.2 The paradigm shift

According to Vayda[32] choosing the right team implies choosing the right attitude rather than technical skills which can be learned. "The most important factor in getting OO technology inserted was skill leverage....the good procedural designers became the good object designers, probably because they had abstract reasoning capability".[17] Graham[11] reports that the main choice most organisations have is between traditional developers with many years of experience and newcomers with skills in OO and modern methods. He reasons that neither will do since they should learn from each other.

A South African retail company agreed that attitude (towards OO) is more important than aptitude, thereby supporting Vayda's argument. A small software company felt that people who are good in C will also be good in C++. In contrast many companies mentioned that COBOL developers will have too many learning curves and that for OO one should rather use new people. A banking institution mentioned that it is easy for developers to use C++ or Smalltalk procedurally and that the only way to solve this is by getting new developers and using the old ones for maintenance.

6.1.3 Training companies in South Africa

External training companies are used in most cases with in-house courses offered for big teams. In two cases companies rely heavily on internal mentoring, where new recruits are assigned to older employees for guidance. A number of companies thought that the training companies often had less experience than the companies themselves did and would therefore rather use individual contractors that are good. The experience has also been that courses on offer are limited, for example, there is insufficient training for CORBA. Training companies are often partisan and force the use of their product rather than addressing the real needs of the client.

In general, companies did not make much use of the services of OO consulting companies - except where individual contractors were employed The high costs involved in employing consulting companies was also cited as a reason for not making use of their services. Consulting companies often focus on certain market sectors and some companies felt that these consulting companies would not understand their culture. The comment "South Africa is driven too much by suppliers who think they are consultants" emphasised the need for product-independent consulting.

6.2 Quality

"Software development processes map the abstract theories of the OO technique into concrete and repeatable actions"[7]

Baer[8] recommends moving to SEI CMM level 3 to maximise reuse. Bernsen[8] reasons that the SEI level can influence the ease and duration of the transition to OO and that the transition can only be as orderly as the CMM level. He therefore feels that companies at level 1 should consider the transition to a higher level concurrent with moving to OO. Korncoff's comment was that large scale reuse goes together with the CMM level. In his article,[32] Vayda stresses the importance of ensuring a quality process, by saying that OO requires the development processes required by the CMM (e.g. version control, metrics, inspections, etc) to be handled very well.

Figure 4 illustrates the clear relationship revealed during the telephonic interviews between quality in South African companies and the awareness of OO. The pie chart on the left indicates that in companies where quality is in place, a large percentage of companies fall in the "Have implemented OO" and "Have used advanced OO techniques" categories.

In contrast, the chart on the right indicates that in companies where no quality is in place, a large percentage of companies have no OO experience or have only heard of OO. During the detailed interviews, however, many companies felt that quality had no influence on the move towards OO. Two cases were mentioned where companies have their own standards and quality procedures already in place even though they have only recently started with OO. In contrast, a company was mentioned where OO has been implemented successfully even though the company is at CMM level 0.

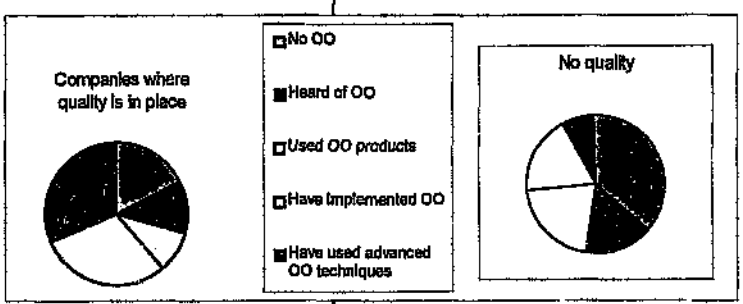


Figure 4OO and Quality

South African companies seem to have the opinion that "if things work leave them because there are other things to do", and that quality is a good thing to have, but, in practice there is no time for it. One suggestion was that organisations should be

"getting a handle" on the OO technology first and change as little as possible at the same time.

More positive opinions, supporting Baer's argument, were that if more processes were in place it would be easier to implement OO (although no CMM level can be a prerequisite for success), and that implementing good OO might even improve the quality process since certain processes have to be followed. A warning came from a retail company against merely having quality in place for the sake of "ticking boxes".

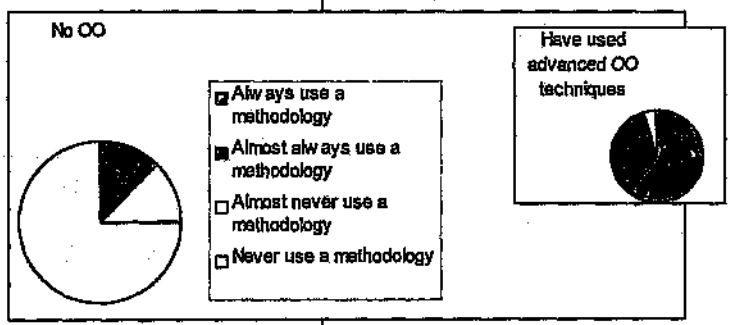


Figure 5OO and methodologies

6.3 Using methodologies

When asked about the importance of methodologies in software development today, Stroustrup[21] replied: "For larger projects, the rules and processes we call methods are necessary, for smaller projects less rigorous approaches are often preferable. Methods are too often used in an attempt to compensate for lack of direction ... and a lack of concepts in the programming used. A method is not a substitute for thinking and understanding. Methods should be applied flexibly enough to accommodate the varying talents, tastes, and weaknesses of a diverse manager, designer, and programmer population." It is seen that methodologies should therefore be used correctly to be of value in the development of OO systems.

According to the Cutter Consortium Report[12], UML (now officially approved by the OMG) is already used by 15% of companies. Most companies indicated a move to UML during 1998. These results were confirmed during the January OMO survey [22], with the primary object methodology used in most companies being UML (43.8%). The phases for which methodologies were used were analysis (24%), architecture (23.5%), high level design (24%), detailed design (19%), and testing (9.5%). Surprisingly, 16.4% of companies did not use any methodology.

Relating this to the situation in South Africa the telephonic interview indicated a clear relationship between the presence of a methodology and the awareness of OO within the company. (Figure 5) The chart on the left indicates that where companies do not use a methodology, there is typically no OO experience present. The chart on the right indicates a clear relationship between companies that have already used advanced OO techniques and the presence of a methodology.

In the detailed interviews, it became clear that, where present, UML was the most popular methodology being used, although most companies created their own customised version of it. Complaints about UML included that it only described the notation and not the process. As with metrics [See section 6.7.2], the recommendation was that no specific process will provide all the answers ("not all of the ticks on the checklists always apply") and therefore the process should be adapted as required.

In most cases the methodology is used for the whole development life cycle, but it seems to be more successful from software analysis to testing. Many companies mentioned that the more involved one becomes in OO, the more one realises the importance of using a methodology. However, the research showed that in many companies no methodology was used, highlighting once again situations where new tools are being adopted rather than the OO methodology *per se*.

6.4 First OO project

For a company tackling its first OO project the following recommendations can be found in the international literature:

- "Mistakes are a normal part of the learning process so ensure that the project can afford to make mistakes and take the time to learn it right. The pilot project should be done in a short interval so that it provides timely feedback. Understanding and applying the technology correctly is more important than meeting the schedule" [16]
- Korncoff[8] recommends choosing a small non critical problem to establish an initial success.
- According to Vayda[32] the project should have high visibility but also a lenient schedule.
- The recommendation is not to tie the budget for introducing object orientation to a single project budget. [23]
- According to Fayad [7] the first project must be a new project which does not have added issues, such as legacy systems.
- The first project must be large and meaningful enough to influence the attitude towards OO of staff involved in other projects. [7]

Comparing this with the results found in the local research, in all but one company, the first OO project was a critical project. This seemed to stem from the nature of the business where the project was often the main project (only project) or where it takes a critical project to get management attention and commitment. Suggestions for the first (ideal) project matched the international literature: a project where developers can first learn everything about OO, that doesn't have impact but that seeks commitment and is of low risk. This suggests an "in between" project. If too small, it will not matter whether it worked or not and people will not know about it. If too big, failure could lead to disaster. The project should be short (six months to one year) because management will want to see results in the short term.

6.5 A difference in management

Regarding the management of OO projects, according to Kenny Rubin from ParcPlace in the USA, the requirement is to appreciate the influence of object technology on software development processes, resources, and products. [13] According to Coplien from AT&T, the differences for OO projects relate to doing risk management in a field of emerging tools and methods that lack the track records available for other methods.[13] Johnson's requirement is understanding both project management and the cost issues, as well as having OO development experience." [13] Fayad[7] found that the manager needs to deal with new or different problems: staffing, training, scheduling, cost estimation, standards, documentation, etc.

From the local research, although it seemed in South Africa that management played an important role in the transition, there was no consensus as to whether the management of OO development is indeed different from the management of other projects. A consulting company found it to be so different that they in fact offer a course on the topic. The opinions expressed did not reflect a clear separation between the experienced and less experienced companies. Arguments given to support the change in management included the short turnaround when using an iterative approach, (in contrast with structured methods where there are long processes) that influences project planning, as well as the paradigm shift resulting in a new breed of programmers to manage.

6.6 Databases

"... small companies are ahead in adopting OO databases, reflecting the stronger hold that database managers in large companies have on their organisations." [12]

In South Africa there was unanimous agreement with the above statement. The reasoning behind this is that in small companies individuals take pride in what they do and are less conservative, whereas in large companies there are more rules (more bureaucracy) and more people that are reluctant to change. Also, due to internal politics and the lack of communication, management in large companies are often uninformed and are influenced by vendors' propaganda as well as myths and prejudice about OO databases being unable to handle large volumes of data. It is also easier and cheaper to change in smaller companies due to less data, whereas large companies cannot afford down time and are restricted by legacy systems. The reality in South African companies is

often having a small number of clients and being forced to use what the client demands, which might not necessarily be an OO database.

6.7 Tools: CASE and metrics

6.7.1 CASE Tools

"While vice-grips may be used to drive in a nail, a craftsman will always use a hammer!" - Baker[2]

The following problems related to the use of CASE (Computer Aided Software Engineering) have been documented by various authors:

- "CASE tools allow poor designers to produce bad designs much more quickly. "- Grady Booch.
- "The continual introduction of new products makes it easy to develop a love/hate relationship with tools", says Shan[31]. The author found that while it is important to be quick to adopt new tools that can help speed up development and improve quality, others must be avoided because of additional training times, unnecessary complexities, and inappropriateness to the job at hand.
- According to Malan[18] CASE tools at the high end are not intuitive to use, whereas tools at the low end provide minimal support.
- Standardisation of tool integration is still not complete which means that CASE tools might not be ready for full time deployment and can therefore cause more problems in large-scale projects. [9]
- Most CASE tools do not cover the full life cycle.[4]
- There is a lack in on-line support to aid the user. [4]
- CASE tools often have unrealistic process requirements regarding the order of specification of a diagram.[20]

South African companies also questioned the state of the CASE tools available at present. Most of these companies therefore do not use any CASE tools. If they do, it is mostly for the documentation of designs and not for code generation. The following problems were raised:

- CASE tools are effective for most of the work but often ignore the last (important) part of the work. This makes the process take longer when using CASE.
- It is very expensive (especially if it only serves as a drawing tool) aimed at the big users.
- There is no time to get to know the tool and related to this, the tool could give you the wrong results if you don't know it thoroughly.
- Experiences are that CASE tools are not flexible enough.
- In the cases where a good CASE tool seemed to be at hand, management could not be convinced to spend the money on something that they felt "they could not see".
- Another comment was that once the (quality) process has been mastered the CASE tool will help, but as a result of having the process, not the CASE tool.

6.7.2 Metrics

Moreau [19] found that very little research has been done towards analytically measuring and quantifying the advantages of OO: *"Unfortunately many existing metrics that have been utilised within conventional programming environments are inappropriate for evaluating OO systems in certain circumstances."*

During our detailed interviews, metrics seemed to be a less important issue for most of the companies. Very few companies used any form of metrics since metrics do not measure up to expectations - the few metrics available for OO being counterproductive, not easily understandable and not usable.

There is also no consistent standard for the usage of the metrics available even if the tools are good. Therefore no comparison can be done with other projects or companies (This seems to be true for all methods and is not only OO related). The recommendation was that companies should look at their own situation individually and not take the tool's output at face value. As with CASE tools,

management often does not see the need to pay for something that is not tangible. At the same time, metrics are important where sceptical management is involved. In small companies, due to priorities, there is no money available for metrics. This is made worse by the fact that metrics are often complex and require one dedicated person. In an insurance company, the perception was that since the company is at CMM level 1, having metrics would be futile since metrics go hand-in-hand with having quality in place.

7. Company profile

In an attempt to describe from the research the typical company that would make a successful transition to OO, the following issues were explored.

7.1 Market sector

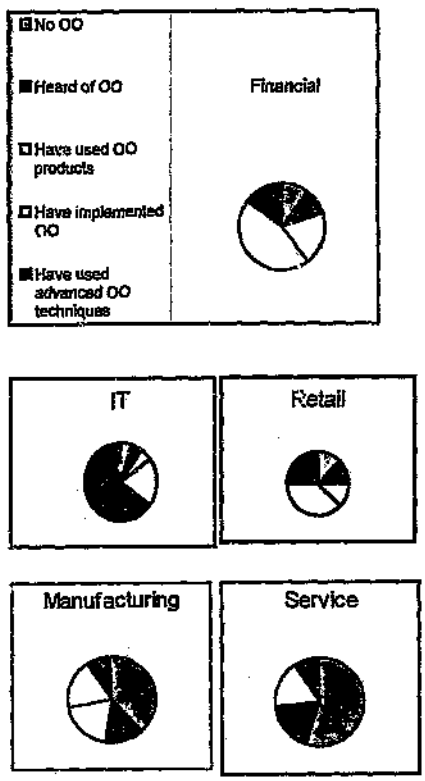


Figure 6 OO and market sector

The telephonic interviews indicated that the retail, finance and IT (Information Technology) sectors are more advanced in the implementation of OO, when compared to, for instance, the manufacturing and service industries (Figure 6). This was also confirmed during the detailed interviews. Financial software is often complex and OO provides a way to handle this. South African banking technology is advanced in international terms and therefore it is likely that the most advanced software development technologies will also be used. The dynamic environment found in the retail, finance and IT sectors implies that new requirements are emerging all the time; OO allows for making these changes. An interesting observation was that although the notion of financial, IT and retail sectors being advanced in OO is true, the competitive nature of these industries necessitates all of their technologies to be leading edge.

7.2 Company size

According to the Cutter consortium report[12], early transitions to OO in large companies have resulted in several failures. However, most IT managers feel confident that they can develop OO applications successfully and large companies are generally ahead of smaller companies in adopting OO.

Vayda's experience [32] in large scale projects was that "applying the OO approach in the industrial setting turned out to be a battlefield strewn with landmines in a number of surprising areas". These areas included different groups that had helped develop the project and therefore had different understandings of the problem, integration with other applications, and platform changes being part of the requirements. Also mentioned was data relationships that are complex and databases that are large, high performance requirements, solutions that have to be scaleable, a lot of inertia and resistance to change, and finally, lots of political issues.

The telephonic interview didn't reveal any significant relationship between company size and the adoption of OO. However, the final interview revealed that smaller companies would generally find it easier moving to OO, motivated by the need for a radical approach to be successful and competitive. Large companies typically have many

software systems developed using other technologies in use, as well as numerous legacy systems to maintain. This requires more co-ordination and communication. At the same time large companies are faced with more people resisting the change. The conclusion is that unlike small companies, in large companies a concept such as OO is not always pervasive and will not be accepted throughout.

7.3 Project size

Although the telephonic interview did not confirm this, the popular opinion amongst South African IT practitioners is that smaller projects will do better in the transition to OO, having fewer people involved with differing opinions, better communication and less requiring change, even though in large companies OO comes to its true value because it scales well.

8. Timing the process

"3-5 years is necessary to change the way in which software is developed as well as acquiring an OO infrastructure." - the Cutter Consortium report.[32]

During interviews, the following were mentioned as factors influencing the speed of adoption of OO in a South African company:

- Age of the developers - young people want to learn, older people are indoctrinated with structured principles.
- Knowledge of the business – as with all methodologies (not just OO) it influences the time frame for success.
- Quality - the importance of the development process
- Technical skills – surprisingly this seemed to be last on most people's list.

The next logical question was whether it is easier for a company introducing OO now compared to those who attempted to in the past, as mentioned in the following statement.

"Object orientation, recently a revolutionary new approach to software, is now joining the mainstream of software techniques. A shop currently embarking upon object orientation is therefore no longer an early adopter and is no longer forced to make a pioneering voyage to Terra Incognita." - Page-Jones[23]

During local interviews, arguments against the above statement were that new tools will not solve the problem and that companies will still face the same basic problems. Problems occur especially where companies have investments in the older technologies. The longer a company waits, the longer it will be stranded with old technology and skills.

Arguments in favour of the statement included the fact that tools, formal methods and training have become available making it practical to implement the technology. Since it has been proven, OO is now an accepted technology even for companies that are not leading edge and where the "critical mass factor" is important[5].

Although no agreement could be reached on this issue, if the time of adopting OO does play a role, it is relevant to find out what the attitude was regarding the future of OO. During interviews, a consulting company reasoned that although the standards are still evolving, the technology is already widely used. The question is whether the technology has reached maturity or is still evolving. If change is inevitable, what will these changes be?

8.1 Is object technology still emerging?

The characteristics of an emerging technology are that *"there is more written about it than known about it , there are more people selling it than using it and the vendors are making more money from education than from selling the tools"*. This still seems to apply to OO.[25]

New developments in OO include the following:

- Next generation object oriented languages
- Security and safety critical software[27] - there is a growing interest in the use of formal methods for safety critical software.

---

[5] The company will not use a technology if the rest of the market does not

- Server objects[30] which will make it possible to separate client design and server design. The important object languages (Smalltalk, C++) either currently support or have planned support for host class server environments. Clients are reaching the boundary of what they can provide without changing the server application. These workstation applications are becoming overly complex and will soon represent the next legacy problem.

- Reengineering OO legacy systems - according to Casais[5], companies that pioneered the move to OO now face the evolution of thousands of classes which represents a new kind of legacy systems. Casais commented that "given the pace at which all economic sectors are taking up OO, an OO reengineering technology is rapidly becoming an acute necessity." His article presents 18 approaches to reengineering OO systems.

- Components. The Cutter consortium report[12] concluded that most companies have started developing some form of component libraries, and 40% of these companies already have frameworks. Further evidence is a recent name change: Object Magazine became Component Strategies as a result of the evolution from object technology to the newest wave in application development.

In South Africa the expectations for the future are as follows:

8.1.1 Tools

According to an insurance house, OO has emerged in the last year but is not mature yet – the measure being the variety of tools still available. It is however exactly in this area (the tools) where most companies see new developments in the future, including fine tuning of the standard bodies and a shift in development tools where you will do less coding.

8.1.2 Components

A viewpoint matching the trend in the literature, came from a consulting company, arguing that object technology was a thing of the early 1990s and that we are already in a component based phase.[6] This also relates to the opinion of a telecommunications company that the technology of the language is mature but that there is now growth in CORBA and the start of true components. Agreement came from a merchant bank, reasoning that emerging areas are component based systems, distributed systems and databases. (The ODMG only published new specifications for databases recently). Therefore the next step for OO will be distributed object standards, standardised component software and pluggable items. The San Francisco project[7] of IBM was also mentioned as the direction where OO is heading.

9. The global situation

In this paper we have described the progress in South African companies towards the adoption of Object Orientation.

The problems experienced were highlighted and some solutions were proposed. One of the questions in the detailed interview aimed to find out what the perceived difference in progress towards OO is locally compared to progress abroad. The demand for South African developers overseas suggests that we have very good technical capabilities, with developers experienced in a wide range of skills. Though fewer, there are islands of excellence here that are comparable to overseas skills.

There is, however, also evidence suggesting that the high levels of skills available in South Africa are not applicable to OO development, perhaps due to a lack of awareness of OO in this country. There is no OO guru (a Booch or Fowler equivalent) in

[6] This is also reinforced by the company's Web page motto: "Leaders in component based technology"

[7] The San Francisco Project is IBM's move towards providing reusability: it delivers on the promise of object-oriented programming by providing a set of server-based application frameworks - it consist of about 1000 object-oriented system-independent class libraries that provides developers with the necessary building blocks for developing server-based applications.[28]

South Africa. There is no support for Smalltalk here. OO has not been implemented as widely here because there was never an urgency to do so.

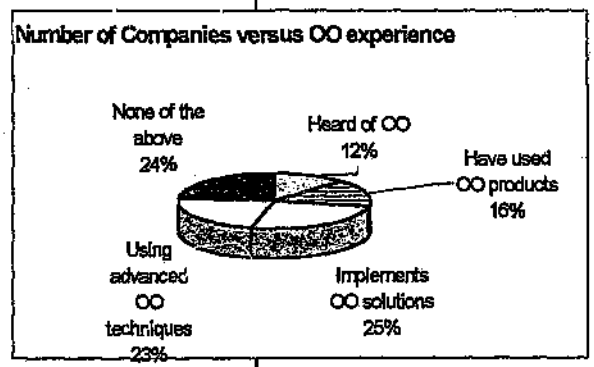


Figure 7OO experience in South Africa

In other countries there are various institutions where OO is already a mature technology and large OO systems are in place. According to a large export company which was interviewed in our research, in product utilisation we are on par with companies overseas but in new development we lag behind.

Finally, a consulting company in the IT sector was interviewed and provided the following information: generally South Africa is 18 months to 2 years behind the US in various technologies. In the US, in 1994, 21% of companies had an OO strategy, in 1995 45% and in 1996 94%. The verdict is that South Africa is at the 1995 mark. Figure 7 illustrates the presence of OO in the 120 companies used for the telephonic interviews: less than half (48%) of the companies surveyed have developed OO applications, which therefore matches this verdict almost exactly.

In conclusion, it is this last comment that suggests that even though South Africa as a whole has excellent technical capabilities, we are still behind in the OO arena. It was also mentioned that this situation might get worse, with young developers, who are the most mobile, leaving the country at an alarming rate.

10. Conclusions

In this article the progress South African companies have made in their transition to OO was described.

In many ways the experiences here match the expectations found in the literature describing experiences in the USA, such as rejection of CASE tools, despondency about metrics and inexperience with reuse. The reasons why many companies move to OO is an area of concern, as well as the lack of quality processes and change in organisational structure. The lack of methodologies in place emphasised the many cases where new tools on the market were adopted and not the OO methodology *per se*. Perhaps due to the specific circumstances here, companies do not have the time or resources available to choose the ideal first OO project. The lack of experienced consulting services contributes to this predicament. Fortunately, it seems as though the expectations South African companies have for the future of OO, is on par with the predictions found in the literature.

11. Reference

[1] Arnold T.R. and Fuson W.A. Testing "In a Perfect World", *Communications of the ACM*, vol. 37, no. 9, Sept. 1994, pp. 78-86.

[2] Baker M. Position Paper for OOPSLA Workshop: Are Object-Oriented CASE Tools ready for Prime Time, http://www.compsvcs.com/cs3/baker.html

[3] Basili V. Briand L.C. and Melo W.L. How reuse influences productivity in object-oriented systems, *Communications of the ACM*, vol. 39, no. 10, Oct. 1996, pp. 104-116.

[4] Benner K. Position Paper for OOPSLA Workshop: Are Object-oriented CASE Frameworks Ready for Prime Time, http://www.compsvcs.com/cs3/benner.html

[5] Casais E. Re-Engineering Object-Oriented Legacy Systems, *Journal of Object-Oriented Programming*, Jan. 1998, pp 45-52.

[6] Corporate and Professional Publishing Group, *Refactoring*, http://www2.awl.com/cseng/titles/0-201-89542-0/Refactoring.htm

[7] De Champeaux D. Balzer B. Bulman D. Culver-Lozo K. Jacobson I. And Mellor S.J. The OO Software Development Process, *OOPSLA '92 Proceedings*, Oct. 1992, pp. 484-489.

[8] De Champeaux D. Baer A.J. Bernsen B. Korncoff A.R. Korson T. and Tkach D.S. Strategies for Object-Oriented Technology Transfer, *OOPSLA '93 Proceedings*, Oct. 1993, pp. 437-447.

[9] Flensted-Jensen N. Position Paper for OOPSLA Workshop: Are Object-Oriented CASE Frameworks Ready for Prime Time, http://www.compsvcs.com/cs3/neils.html

[10] Graham I. In Search of the Three Best Books *Journal of Object Oriented Programming*, Sept. 1997, pp. 43-45.

[11] Graham I. Migrating to object technology, Addison-Wesley UK, 1995, pp.3-72, pp.399-444.

[12] Harmon P. The corporate use of object technology, Cutter Consortium, http://www.cutter.com/itgroup/reports/corpuse.html 1997.

[13] Hill L. Rubin K. Daniels J. Berman C. Coplien J. and Johnson D. Managing Object Oriented Projects, *OOPSLA '95 Proceedings*, Oct. 1995, pp.88-90.

[14] Korson T, Managing the Transition to Object-Oriented Technology, *OOPSLA '91 Proceedings*, Oct. 1991, pp. 55-62.

[15] Kristek T. and Vaishnavi V. Role of a Corporate Technology Center, *OOPSLA '94 Proceedings*, Oct. 1994, pp. 72-77.

[16] Lato K. Learn to learn: Training on new technology, *Journal of Object Oriented Programming*, Mar/Apr. 1997, pp. 24-27.

[17] Litvintchouk S.D. Evolving Towards Object-Oriented Technology In Large Organizations, *OOPSLA '93 Proceedings*, Oct. 1993, pp. 73-76.

[18] Malan R. Coleman D. and Letsinger R. Lessons from the Experiences of Leading Edge Object Technology Projects in Hewlett-Packard, *OOPSLA '95 Proceedings*, Oct. 1995, pp. 33-46.

[19] Moreau D.R. and Dominick W.D. Object-Oriented Graphical Information Systems: Research Plan and Evaluation Metrics, *The Journal of Systems and Software*, vol.10, 1989, pp.23-28.

[20] Narayanaswamy K. Are Object Oriented CASE Frameworks Ready for Prime Time, *OOPSLA '95 Proceedings*, Oct. 1995, pp. 155-158.

[21] Object Magazine Online, The Object Magazine Online Interview -Bjarne Stroustrup, 1998, http://www.objectmagazine.com

[22] The Object Magazine Online Survey: What are you using?, Jan. 1998, http://www.sigs.com/omo/questionnaire/results.html

[23] Page-Jones M. Object Orientation: Making the Transition, copyright Wayland Systems Inc. 1997, pp. 1-19.

[24] Page-Jones M. The seven stages in software engineering, *American Programmer*, Jul/Aug. 1990.

[25] Pancake C.M. The Promise and the Cost of Object Technology: A five-year forecast, *Communications of the ACM*, Oct. 1995, vol.38, no.10, pp.33-49.

[26] Reed D.R. Cagan M. Goldstein T. and Moo B. Issues in Moving from C to C++, *OOPSLA '91 Proceedings*, Oct. 1991, pp. 163-165.

[27] Riehle R. Reuse OOP and Safety-Critical Software, *Journal of Object Oriented Programming*, Nov/Dec. 1997, pp. 21-24.

[28] Scannell E. IBM rolls out building blocks for San Francisco Project apps, Infoworld, vol. 18, issue 52/53, Dec. 1996, http://www.ibm.com/Java/Sanfrancisco/

[29] Schach SR. Classical and Object-oriented software engineering, third edition, Irwin/McGraw-Hill, USA 1996, pp. 70-74.

[30] Shan Y-P. Morgan T. Proudfoot P. Thompson J. Tibbetts J. and Woolfrey A. Objects on the Server: Are We Ready, *OOPSLA '96 Proceedings*, Oct. 1996, pp. 384-388.

[31] Shan Y-P. Auer K. Bear A.J. Adamczyk J. Goldberg A. Love T. and Thomas D. Smalltalk in the Business World the Good the Bad and the Future, *OOPSLA '94 Proceedings*, Oct. 1994, pp. 145-152.

[32] Vayda T.P. Lessons From the Battlefield, *OOPSLA '95 Proceedings*, Oct. 1995, pp. 439-452.

[33] Wick M.R. On using C++ and Object-Orientation in CS1: The message is still more important than the medium, *SIGCSE '95*, Mar. 1995, pp. 322-326.

12. Author Contact Details

Contact details: Miranda Jansen van Rensburg, Technikon SA (Information Technology), C de Wet Drive Roodepoort Office Phone: +27-11-471-2929, Fax: +27-11-471-3270, Internet E-mail: mjvanren@tsamail.trsa.ac.za

Pitfalls and guidelines in the transition to Object Orientation

A global view

M Jansen van Rensburg

Software Engineering Applications Laboratory, Electrical Engineering,
University of the Witwatersrand, Johannesburg, South Africa

Summary:
This article provides a description of the research done for a MSc research report. The study looks at the pitfalls and guidelines in the transition to object oriented software design methodologies in South African companies and compares the results with those obtained in the USA. The results reveal certain similarities and differences as well as areas of concern, where neither South African nor USA companies have made progress in the transition.

1. Introduction

This article describes the final chapter in a research report investigating the pitfalls and guidelines in the transition to Object Orientation (OO). A previous paper described the South African situation in this regard. This article however aims to compare the local situation with the global one, to find out whether South African companies are indeed facing the same problems and, if similar, whether the methods of addressing these problems are akin. The paper is presented from a South African perspective and the term "local" used in numerous cases therefore describes the South African situation.

The last section describes possible areas where both the local and international companies are lacking, and that are therefore areas of concern.

2. Data collection

Firstly, a literature survey was done to find out what is already known about this research area. At the same time a few informal interviews were held with selected South African companies to determine what the issues involved are.

Data was collected in South Africa by means of telephonic interviews with 120 companies from various market sectors, as well as more detailed interviews with a selected number of companies.

Sources of information regarding the global situation included in depth interviews with members of companies abroad, as well as a presentation done by an overseas company in South Africa. Again this selection aimed to represent various market sectors, the final selection including the information technology, manufacturing and finance sectors. Both large and small companies were interviewed. In contrast with the situation in South Africa, an abundance of information and experiences regarding companies abroad was available. Therefore fewer companies were interviewed in the USA than in South Africa.

3. Initial benchmarking

3.1 A first glance

The international situation can be quantified as follows:

Pickering's survey[45] shows that the situation regarding OO in 1993 could be described as given in Table 3.1:

Table 3.1Effective Penetration of OO

Projects used (%)		Success (%)		Effective penetration (%)	
1991	1993	1991	1993	1991	1993
3.8	11.9	91.7	66.3	3.5	7.9

In recent interviews held with Application Development Managers, the Standish Group[34] found that OO programming was a highly significant factor in choosing an application development tool for 81% of participants. (Figure 3.1)

This is confirmed by the Cutter Consortium report[21] for which more than 200 enterprises worldwide were used in a survey. Between 75% and 80% of companies is either exploring or using object technology.

In comparison, results from the telephonic interview conducted with 120 South African companies, summarise the local situation as given in Figure 3.2. Less than half (48%) of the companies surveyed have developed OO applications.

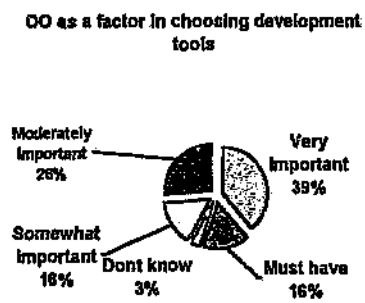


Figure 3.1 OO in the USA

3.2 What South Africans thought

During local interviews, the question posed to companies inquired about the perceived difference in progress between local companies and those abroad. The result was the following:

The demand for South African developers overseas suggests that South African companies have very good technical capabilities, with developers experienced in a wide range of skills. Though fewer, there are islands of excellence here that are comparable to overseas skills.

The general opinion was that the good capabilities we have here are not applicable to OO development, perhaps due to a lack of awareness of OO here. The number of people attending conferences here compared to other countries confirms this. There is no support for Smalltalk here.

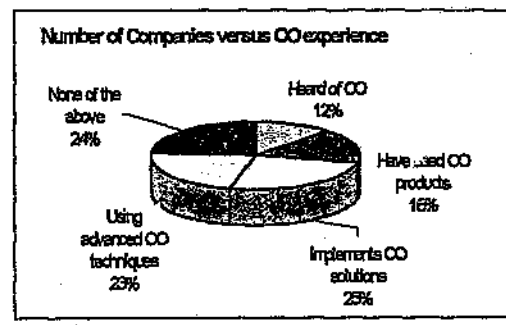


Figure 3.2 OO experience in South Africa

"Overseas there are various institutions where OO is already a mature technology having large OO systems in place." According to a large export company, in product utilisation we are on par but in new development we are behind.

3.3 Some perceived differences

From the results obtained locally, the following were thought to be the perceived differences from the global situation:

- There is no time in South Africa to experiment or investigate new tools which means that the first project is most often not the ideal non-critical project, whereas in the USA there might be more leniency to allow for experimentation.
- Because of skill shortages and a lack of training, South African companies will only support major technologies, since it is too

expensive to diverge. If OO is therefore not considered to be used by the critical mass, it is not supported.

- Due to high exchange rates, South African companies can often not afford the luxuries of purchasing expensive CASE tools. Perhaps this problem is not occurring in the USA.

- Many South African companies think they are implementing OO, but are in fact just using the tools. It is not clear whether this also occurs in the USA.

- There has been close to no change in the organisational structure of South African companies adopting OO, in spite of various papers suggesting such a change. Many of these papers were written by USA individuals which suggests that perhaps this problem does not exist in the USA.

4. A comparison

After data collection was completed, results obtained locally and abroad were compared:

4.1 General comments and problems

In the 1997 Cutter Consortium's report[21], 41% of companies reported at least one OO failure. It is therefore worthwhile exploring some of the problems and successes that were experienced both locally and abroad.

A survey[45] summarising reasons given by 1991 and 1993 survey respondents for NOT using OO at the time, found the reasons to be as given in Table 5.1.

Table 5.1 Reasons for not using OO

REASON	1991	1993
Not aware of technology	31.0	13.2
Benefits not demonstrated	3.5	19.3
No business need	17.2	3.5
Technology too costly	0.9	2.6
Organisation unprepared	19.8	19.3
Technology too immature	19.8	36.8
Other	7.8	5.3

A survey was done of the members of the Connecticut Object Oriented User Group (COOUG) in June 1995 on the "state of OO" in the area (being the greater Connecticut / New England area in the USA)[6].

The survey explored areas that were the most difficult in working with object oriented technology and found the following:

28% - finding good OO people

27% - designing for reuse

20% - project management

According to a case study done in the Travelers Group in the USA[27] the following difficulties were experienced:

- conceptual difficulties (the paradigm shift)

- technical difficulties (new methods and tools that are immature and lacking standards)

- organisational difficulties (system designers finding it more difficult to understand existing components than to create new ones and also believing that "if not invented here" , systems are not good enough.)

- Political difficulties (securing funds is required since building for reuse can cost more). In the early 1990s, insurance suffered from over-capacity and therefore competition was rife.

The guidelines for solving these problems were reported to be the following:

- Following an iterative process in development.
- developing in-house expertise, creating centralised support,
- developing a three-tier architecture to allow developers to specialise in a limited number of technologies
- political problems were solved through better communication with business partners
- OO development in the pure sense can not be implemented practically; Business needs change too quickly so the time required to fully develop the OO model will never be available.
- accepting that technologies are unstable and changing – this emphasises the need to have objects that allow you to change between platforms easily
- the fact that new systems and immature technologies require adjustment to organisational and technical problems simultaneously, needs to be accepted and dealt with
- adopting a learning attitude that recognises the need to learn from mistakes
- designing a reliable cost effective architecture for supporting the OO environment
- handing over ownership for technology problems

During the local interviews, the problems that companies were experiencing were the following:

- Lack of proper design, and not using a methodology; The importance of the methodology was often not stressed sufficiently
- Lack of skills, coupled with having no time for training
- Not having the luxury of experience gained on a smaller project
- Not having the luxury to investigate different tools first
- Many companies are still operating at a SEI CMM[1] level of one, thereby having hero programmers – if these programmers leave there is no one to take over
- Lack of full time project management
- In the case of a large merchant bank, using a relational database – in this case 30% of the code and 60% of the performance were compromised just to handle the conversion from a relational to an OO database
- Lack of proper documentation
- Being too calendar driven – deadlines have to be met regardless of the quality of the system being developed
- Lack of communication amongst users and between users and developers
- The paradigm shift, requiring a new way of thinking
- At a large defence industry company it was found that despite using OO for handling complexity, the system analysis was still functional in nature. It was difficult to find a transition from analysis to design. There was no automatic process for integration of the whole life cycle

4.2 Time when adopting OO

A South African consulting company in the IT sector provided the following information: generally South Africa is 18 months to 2 years behind the USA in various technologies. In the USA, in 1994, 21% of companies had an OO strategy, in 1995 45% and in 1996 94%. The verdict given was that South Africa is at the 1995 mark. This is confirmed by the telephonic interviews result provided earlier, stating that 48%

[1] The CMM (capability maturity model) was developed by the SEI (Software Engineering Institute) at Carnegie-Mellon University, and defines five levels of maturity in the software process of organisations.[50]

of South African companies surveyed have developed OO applications.

This also agrees with results from the 1995 COOUG survey: The satisfaction index with OO seemed to be mediocre at best with the majority of companies giving it a 2 out of 5.Most companies were just sta ting with OO and found it too early to see benefits. A few companies were however totally committed to OO finding it essential to the way they develop applications.

The impression is therefore that South African companies are at least two years behind their USA counterparts. Surprisingly, however, the companies interviewed in the US started their adoption of OO at the same time as companies in South Africa, averaging at about 4 years ago.

4.3 Reasons for choosing OO

There are numerous sources available describing the advantages related to OO, including:

- Productivity through reuse[21]
- Higher quality systems[40]
- Higher quality development process [40]
- Capacity to build larger systems[11]
- Rapid development: Sound object oriented analysis helps to deliver the benefits of OT (Object Technology) much earlier in the life cycle[33]
- Encapsulation[33]
- Extensibility and flexibility [33]

Although these conventional (correct) reasons for choosing OO were mentioned during local interviews, some interesting reasons for moving to OO were mentioned on numerous occasions in South Africa:

- improving quality
- "All the new technologies are in OO" - there is a need to move with the new technologies available
- Developers' needs – developers do not want to work on mainframe systems. Companies want to keep the right people and therefore are forced to move to the new technologies

The last two reasons mentioned are issues of concern, as Page-Jones classified both as being the wrong reasons for adopting OO.[43] However, in South Africa, due to skill shortages, it seems that OO is used as the proverbial new technology carrot that will prevent developers from leaving the company, by satisfying developers' needs for having skills in the latest technology.

Without exception, the reasons for choosing OO mentioned by the USA companies interviewed were the so-called correct reasons such as handling complexity and achieving maintainability. Reuse was explicitly found not to be a reason for choosing OO and was also not a reason for pursuing OO in any of the South African companies.

4.4 First Project

Various opinions regarding the desired characteristics of the first OO project in a company are expressed in international literature:

- The pilot project should be done in a short interval so it provides timely feedback
- "Understanding and applying the technology correctly is more important than meeting the schedule" – Lato[28]
- Page-Jones [42] recommends accepting that the first project will not yield great financial dividends, and might wind up *costing* money. The project team will take time to learn, and new tools for development and library management will require a monetary investment. He cautions against building up unrealistic expectations and against trying to adopt every last aspect of object orientation all at once
- According to Vayda[55] the project should have high visibility but simultaneously also a lenient schedule
- The first project should be large and meaningful enough to influence the other project members' attitude towards OO [15]

Comparing this with the results found during the South African interviews, the first OO project was almost always a critical project. This seemed to stem from the nature of the business where the project is often the main project (only project) or where it takes a critical project to get management attention and commitment. Suggestions for the first (ideal) project matched the literature: a project where developers can first learn everything about OO, that doesn't have impact but that seeks commitment and is of low risk. The project should be short (six months to one year) because management would want to see results in the short term.

Also in the USA, relevance and high visibility were mentioned to be the critical factors. The project should therefore be sponsored with a lot of political influence. Surprisingly, as in South Africa, in the Travelers Group the first project was a large application of high visibility. This was however considered positively since development had to continue at a time when developers, who had no prior OO knowledge, were ready to quit.

4.5 Methodology

According to the Cutter Consortium Report[21], UML (Unified Modelling Language), now officially approved by the Object Management Group (OMG), is already used by 15% of companies. Most companies indicated a move to UML during 1998. These results were confirmed during the January OMO survey [39], with the primary object methodology used in most companies being UML (43.8%). The phases for which methodologies were used were analysis (24%), architecture (23.5%), high level design (24%), detailed design (19%), and testing (9.5%). Surprisingly, 16.4% of companies did not use any methodology.

During local interviews, it became clear that, where present, UML was the most popular methodology being used, although most companies created their own customised version of it. South African companies were more critical than USA companies in their evaluation of UML, complaints being that it only described the notation and not the process. The recommendation was that no specific process would provide all the answers ("not all of the ticks on the checklists always apply") and that the process should be adapted as required.

Still, in many companies no methodology was used, highlighting once again situations where new tools on the market were adopted and not the OO methodology *per se*. None of the companies interviewed in the USA showed any signs of this problem – they were all well aware of the methodology that goes along with the change.

In the USA, UML is used or alternatively, a "home-grown" combination of UML and others (Rebecca Wirfs-Brock, Booch / Rumbaugh) in companies where UML was not available at the time of choosing the methodology.

The significance of the introduction of UML is hereby noticed since, in the COOUG survey (1995), it was found that numerous methodologies were used, whereas now, globally, there seems to be a convergence towards UML.

The incremental approach in using the methodology is also considered very important in the USA and is implementing correctly. Ron Calabrese, technical director in the Travelers Group and chairman of the COOUG, commented that in the Travelers Group the usage of multiple iterations was very critical to their success.

4.6 Architecture and Organisational Structure

Anderson [1] mentions the importance of management commitment to an architectural approach rather than churning out code, to foster increased architectural competency.

Both local and USA interviews revealed an awareness of the importance of the architecture. As a general guideline to implementing OO, Dennis Laibson (ObjectGems in Virginia, USA) recommended using a wise architect to do a true evaluation of the system, and investing in the architecture first. Similarly, according to EIKON, a South African consulting company, the architecture and a controlled development process should be in place before any other decisions are made.

Just as the architecture is important for the new organisation of the system, this change must also be reflected in the organisation of the people for the successful implementation of OO:

- At BNR they learnt the following valuable lessons[33]: high cohesion and coupling of objects can be used to one's advantage in

organising the people as well as the software but care must be taken to ensure than the organisation mimics the architecture rather than vice versa.

- Johnson [23] reasons that good OO systems are structured differently than conventional systems and therefore a different organisational structure is needed. He recommends teams owning groups of related classes and individuals within teams owning classes.

- Korson[24] claims to have seen "projects fail to achieve the promise of OOT because the organisation did not understand the changes in corporate infrastructure that were necessary to support the object paradigm".

In half of the cases studied locally, the organisational structure has not changed at all. Most thought that such a change was necessary and would happen in the future. The perception was that there are different roles involved in OO as compared to procedural development, such as a system architect who can act as mentor, a project manager, developers and a quality assurance team. In the cases where the structure has changed, it seemed to be related to a general company restructure and not related to OO specifically. It is therefore clear that the organisational change related to OO is an area South African companies need to address urgently.

In the USA, there seems to be a greater awareness of this change in organisational structure.

Calabrese sees a definite need for such a change - in the Travelers Group it took place as follows:

In 1992, the team consisted of 1 large development team (24 developers, including 10 consultants) project managers, 1 senior OO consultant, 1 client server engineer. Compared to this, in 1997 the structure consisted of multiple small development teams (3-5 per team) and project management being done by the team.

For the development of frameworks they had two kinds of developers:

- A business developer, responsible for developing use cases, determining component services and developing business objects and front ends.

- A technical developer, concentrating on technical aspects, designing and developing base classes and mentoring business developers.

They also had a common business object development team concentrating on reuse

4.7 Quality

"Software development processes map the abstract theories of the OO technique into concrete and repeatable actions"[15]

According to a survey amongst panellists at an OOPSLA conference,[30] when asked whether there a relationship between the migration to OT and "process improvement" and the capability maturity model, it seems that the majority of panellists felt that the "process is still more fundamental than which methodology you choose." It is felt that one should rather then use structured methods with a well-defined process than use OO programming languages such as C++ and Smalltalk and "hack".

Although the NASA Software Engineering transition to OT took 7 years, this is typical in any *ad hoc* environment.

The necessity of having an appropriate CMM level for success in OO is discussed in the following papers:

- According to Baer[13] it is not necessary to achieve some CMM level to move to OO, however he recommends moving to level 3 to maximise reuse.

- Bernsen reasons that the CMM level can influence the ease and duration of the transition to OO. However, the transition can only be as orderly as the CMM level. On the other hand high CMM levels could cause difficulty as the company has set practised processes which may lead to inertia to change. He therefore feels that companies at level 1 should consider the transition to a higher level concurrent with moving to OO.

- Korncoff's comment was that large scale reuse goes together with the CMM level.

- Vayda[55] confirmed again the importance of ensuring a quality process, saying that OT requires the development processes required by the SEI (e.g. version control, metrics, inspections, etc.) to be handled very well, regarding consistent analysis, design and documentation, change management, and traceability through the system development life cycle.

- Baer[13] said that although lots of companies have small pockets of OT, when it comes to using it on a large-scale, using objects requires understanding how they impact on every facet of the system development life cycle which again enforces the drive towards quality.

- Although there is an overwhelming *proliferation* of object-oriented development methods, for most companies, even a middle-of-the-road method represents a quality improvement over their erstwhile software practices. This unexpected advantage, namely improving the *quality* of the development process, also leads to an improvement in the *productivity* of development.[40]

Local telephonic interviews illustrated a clear relationship between quality in the company and the implementation of OO - where a quality improvement process was present, the progress towards OO was far advanced. However, during the final interviews, many companies felt that quality had no influence on the move towards OO. Two cases were mentioned where companies have their own standards and quality procedures already in place even though they have only started with OO. In contrast, a company was mentioned where OO has been implemented successfully even though the company is at CMM level 0.

South African opinions included a perception that ISO 9000 certifies the process, not the software - the policy being that "if things work leave it because there are other things to do", and that quality is a good thing to have, but in practice there is no time. One suggestion was getting a handle on the technology first and changing as little as possible at the same time.

More positive opinions, supporting Baer's argument, were that if more processes were in place it would be easier to implement OO (although no CMM level can be a prerequisite for success), and that implementing good OO might even improve the quality process since certain processes have to be followed.

A warning came from a retail company against merely having quality in place for the sake of "ticking boxes".

While the importance of such a quality process is still questioned in some companies in South Africa, the situation is much worse in the USA:

ISO9000 did not seem to play any role in the companies interviewed. Respondents were greatly opposed to the suggestion that the transition should first be to a CMM level e.g. level 3 before implementing OO.

4.8 CASE

The usage of CASE tools in OO projects should not be studied in isolation. Unfortunately, CASE tools in general have been associated with the so-called Fad of the Year phenomenon by numerous authors: Page-Jones[40] reports that "this occurrence (the fad) has seen many shapes: In 1984 The Fad was "relational stuff"; in 1986, artificial intelligence; in 1988, CASE tools ..."

Pickering's survey[45] describes the poor penetration of CASE into the market in 1993. (Table 4.1.)

Turning back to OO, the need for CASE tools seems to be even more critical. Narayanaswamy[36] forecast that OO methods will see increased usage in the future, but that without computerised support in the form of case tools, it will be impossible to contemplate OO technology in large projects.

In 1995, reports came from Hewlett-Packard[31] that the OO CASE tool situation was not satisfactory. Most users regarded CASE tools to be very important. According to Berg's 1995 report of the OS/400 OO project,[4] CASE tools were mostly used to capture designs and the white board for producing the design.

The principal goal of the Aaron project[37] at the University of Technology in Sydney was therefore to ascertain how CASE tools could effectively

support the managed development of quality OO software. The change in requirements for OO CASE tools includes the need for the ability to build share and reuse components across projects,[48] as well as keeping consistency across views in large OO models.[56]

Most South African companies questioned the state of the CASE tools available at present. Most of these companies therefore do not use any CASE tools and if they use any it is mostly for the documentation and capturing of designs done on e.g. a white board. The following complaints were given:

- CASE tools often help with the first round and then when the design changes the whole process has to be redone.
- CASE tools work fine for 80% of what you want to do and for the last part you are left on your own – this is also probably the most important part.
- It is very expensive, aimed at the big users and companies cannot justify it although it is a nice concept. Another company also complained that for a drawing tool it was very expensive and that they wouldn't mind paying if it supported the real engineering process.
- The design process takes longer when using CASE. There is also no time to get to know the tool. Tool can give you the wrong results if you do not know it thoroughly.
- CASE tools are not flexible enough.
- In the cases where a good CASE tool seemed to be at hand, management could not be convinced to spend the money on something that they felt "they cannot see".
- CASE tools help when you have already mastered the process but that is as a result of having the process, not the CASE tool.

Table 4.1 Pickering's survey - CASE tools

Projects used on (%)		Success (%)		Effective penetration (%)	
1991	1993	1991	1993	1991	1993
28.8	26.3	59.7	70.8	17.2	18.6

While these companies still questioned the CASE tools available, USA companies' views were much more vehement:

No CASE tools are used.

The California based company Object-Z, specialise in OO COBOL, for which there are no tools available. Will Price (Object-Z) used Page-Jones's sentiment to describe the situation: "10 years ago that (CASE) was going to solve all the problems." According to Laibson, the development tools have become so powerful that there is no need for CASE tools anymore. Calabrese reported that no CASE is used since the tools are limiting - there is no flexibility from a specific vendor.

Lastly, the weakness in this market is evident from the COOUG survey: 76% of companies interviewed were not using CASE tools. Where CASE was used, there was a variety of more than 10 tools being used – not one tool stood out.

Whether CASE tools will play a role in the future of OO remains to be seen. One South African suggested: "rather leave out the CASE tools and get key people"".

4.9 Metrics

According to Watts Humphrey[42], an organisation's sophistication goes through five levels, or ages, of software-engineering maturity in applying OO. The Age of Metrics is the fourth stage and introduces quantitative measurements of processes. Such metrics is important when approaching management on OO: pure technical aspects will not succeed, one has to demonstrate improved productivity. [24]

Unfortunately, not much information is available on metrics for OO.[24] Metrics are also

fundamentally different for OO. Traditional software metrics do not provide for measuring OO concepts such as classes, inheritance, encapsulation etc.[8]

Locally, metrics seemed to be a less important issue on the minds of most of the companies interviewed. Very few companies used any form of metrics.

The feeling was that the metrics available do not measure up to the expectations. Complaints were:

- There are very few metrics available for OO.
- What is available for OO is counterproductive.
- It is not easily understandable and usable.
- Attention paid to metrics should be related more to the business needs rather than specifically to OO: speed, flexibility and whether it is what the client wants is still more important.
- There is no consistent standard for the usage of the metrics available even if the tools are good. Therefore no comparison can be done with other projects or companies.

  as the case is with CASE tools, management often does not see the need to pay for something that is not tangible. This was mentioned in a large retail company, which suggests that the situation in smaller companies is even worse.
- In the case of an insurance company, it was felt that the company is at CMM level 1, using hero programmers, and that metrics go alongside with having quality.

Guidelines for using metrics were the following:

- metrics are important, especially where sceptical management is involved.
- one person should be dedicated to metrics since it is complex.
- There are good tools available but you should look at your own situation individually and not take the tool's numbers at face value e.g. if the tool says a score of 10 is bad you might get 15 and it might be good for your situation.

In the cases studied, metrics seemed to be even more non-existent in the USA. Arranga (Object-Z) explains this by saying that people are weary of metrics because of the great potential for misuse. Generally he found metrics to be satisfactory but that management have to be trained to use them.

4.10 Reuse

"Sharing is one of the primary benefits in object programming." – Shaw[53].

"Most companies have started developing some form of component libraries, and 40% of these companies already have frameworks" - the Cutter Consortium's report[21]

There is currently little empirical information on what to expect from reuse in terms of productivity and quality gains. [2]

Having a domain analyst that can co-ordinate with domain experts on different projects, develop specifications and designs for reuse and create or maintain a class library is recommended.[16] Reuse is often not well understood and easily oversold. Most classes are not reusable outside the system they were developed for. Also, most companies lack the organisation and communications to support reuse - Reed Philip[24], Knowledge Systems Corporation.

Designing for reuse is difficult for the following reasons:[20]

- Over-generalisation leads to unbearably high infrastructure costs
- The inclusion of application specific ideas in a library intended for general use, leading to restrictions and delay in subsequent projects

Korson[24] emphasises the importance of having proper tools to retrieve and store components, as well as providing rewards for developers submitting reusable classes to the class library and for developers reusing classes.

The Ovum report[41] highlights the role of management arguing that achieving reuse lies more

in the hands of management and organisation, than in a particular technology.

Page-Jones[41] provides advice, saying that formal policies for entering, storing, retrieving and removing library holdings have to be established. He recommends appointing a librarian to oversee the activities of entering, storing, retrieving and removing code. This librarian could be a single person or a small team.

Locally, being a difficult process, reuse was not a primary objective for many companies in South Africa. Reuse is often tied to the kind of business, and development done is therefore often too specific rather than too generic[3]

In the case of the military company where the software is embedded and the hardware changing, it is impractical to try to reuse. A consulting company proposed promoting maintainability and risk management rather than reuse. Other suggestions were using refactoring[2] after development, writing a framework and making that specific and concentrating on having the same development process throughout. According to a large insurance company, reuse comes from repetition, going through multiple abstractions, and not appearing up front in the design. Trying to design for reuse is considered as premature design.

The message is therefore to concentrate on use rather than reuse. Most of these companies are investigating appointing a person as a "reuse miner" to manage the libraries, plan the repository, etc. This person will play an important role since accessibility is a problem if there is no way to browse the objects already developed. The need for a change in the reward structure is also evident: people like to build systems themselves - therefore the reward structure should change to promote reuse, as earlier mentioned by Korson.

In the USA, as in South Africa, reuse was not the primary goal in the adoption of OO. The COOUG survey showed that 75% of companies did not have a central area responsible for managing reusable class libraries, and that 95% did not have a reuse incentive program.

According to Arranga, although huge management is required to achieve reuse, components are going to be the unit of reuse in the future. Price, author of OO COBOL, mentioned during his interview that the value of reuse is overstated. "Reuse is hard to do, you need a lot of insight into the system being developed and the actual business system is complicated."

In the Travelers Group, Calabrese found that in their case the most benefit from OO came in development time of new systems through the reuse they achieved. The new system made it clear to people what was expected to do and created a powerful organisation. It also gave them technology none of the competitors had. The new system lead to reduced costs and higher reliability. However, achieving this reuse milestone was not without problems:

- Some business units did not want to fund something of unknown value.
- Managers did not want reuse considerations to delay delivery,
- "it's very much like an uphill battle. It is funded but its continually being challenged and checked"
- When milestones were not met, reuse requirements had to be relaxed to keep up.

Their reuse directive was to establish one divisional architecture, to not over engineer any component and to create common functions and pluggable components. Development cost went from $7million (1993) to $0.5 million (1996).

4.10.1 From class libraries to frameworks

In general, it seemed as though USA companies might be ahead in the reuse arena, with frameworks being developed successfully.

[2] Often when changing code, additions become very complex but there is no time for redesign. Refactoring describes the techniques that reduce the pain of redesigning. The functionality of the software is not changed, rather the internal structure. It normally involves small steps at a time, such as moving a field to another class. Also, while refactoring no new functionality is added. The technique is still new and is mainly used in the Smalltalk community but promises to improve software development in all environments.[10]

In Hewlett-Packard[31] in 1990, reuse meant class libraries only, whereas in 1995 several divisions were using OO to successfully develop domain specific frameworks for product families.

Also within the Travelers Group[27], frameworks were developed during 1994 to 1996, which was advanced for the time. Their emphasis changed from "learn OT and provide new functionality quickly" (1992), to "reusing wherever possible by promoting a flexible architecture and adding frameworks to support all applications" in 1997.

A three-tier model was developed which was later reused on new projects. The new model had business objects at the top, with parallel objects in the second layer. These frameworks acted as object request broker for queuing of requests. The bottom tier was platform objects that interfaced to specific platforms. Reuse also happened in terms of learnt skills and centrally provided support infrastructure.

4.11 Resistance

Often organisations "undertake OO technology ...without much management support. Because there isn't an atmosphere fostered to make it happen people get resistive attitudes. That causes failure." [30]

Vayda's experience[55] was that there is a lot of inertia and resistance to change to OO in large projects.

In South Africa, in the companies where OO was implemented with a clear understanding of the paradigm shift and the new design methodologies, (i.e. full OO) the process was met with a lot of resistance. The resistance seemed to stem from the following:

- It seemed to be the older developers who did not approve the move to OO
- Often being uninformed, developers, users and managers thought of OO as vapourware (a lot of hype but no results).
- At the time of the transition, developers did not understand the technology and thought it was too complex. Developers typically did not see the need to change
- According to a large commercial bank, some people do not want to change; they feel safe with what they know and what they know is enough to ensure work for lots of years to come. There should be a willingness to get it wrong and try again.

Where there was no resistance, it seemed to be a result of either knowing the technology, (which implies appointing people who are open to change) or due to an ignorance where people were not really aware of what the new process entails. This also relates to the earlier observation that people are only using the tools, and not the methodology. This reality reinforces Orr's [44] observation in 1995, that it is not clear how many people who think they are doing OO programming, actually do OOP, and that it seems to be a small percentage.

In the USA, perhaps due to more exposure to conferences, there was less resistance. In fact, Arranga would rather call it "puzzlement with the unknown".

However, Price found that in the case of COBOL, specifically, people are slow to change and do not appreciate the technical aspects of this new adoption. According to him, not many COBOL programmers know other languages or programming theory as such. In short, the backgrounds are different for COBOL vs. C programmers, which will influence their being resistant or not.

Laibson explained the lack of resistance by saying that the concept (OO) is often loosely interpreted, which again leads us back to the case where many companies think they are implementing OO but are actually just using the tools, as seen in South Africa.

4.12 Company Profile

While trying to establish guidelines for the successful OO project, it is worthwhile to also define the profile of a company that will be able to make a success of the OO adoption.

4.12.1 Market sector

Local research (both telephonic and detail interviews) indicated that the retail, IT and financial sectors are ahead in the adoption of OO. Reasons given included that financial systems are

complex and that OO provides a way to handle this, that the high standard of banking systems in South Africa require new systems continually, and that these sectors are advanced in adopting all new technologies, not only OO.

This result was also echoed during the USA interviews. The following serves as evidence:

- According to Edmund Arranga, there are more pressing needs in the financial sector, whereas the manufacturing sector lags, being more conservative.
- Dennis Laibson sustained this by indicating that the majority of his company's clients are from the financial sector. Since his company consults in OO, it is therefore logical to derive that the financial sector is indeed ahead in trying to implement OO in projects.
- "claims processing is a paper intensive process making IT a critical tool in the industry. IT offered firms the chance to deliver new services faster than the competitors."[27] - a comment regarding the OO initiative within the Travelers Group that operates in the insurance industry.
- Lastly, these arguments can also be quantified by including the COOUG survey:[6] 50% of respondents came from the insurance / financial sector, 14% from manufacturing, 12% from consulting (IT).

4.12.2 Company size

According to the Cutter consortium report[21], early transitions to OO in large companies have resulted in several failures. However, most IT managers feel confident that they can develop OO applications successfully and large companies are generally ahead of smaller companies in adopting OO.

Vayda's experience [55] in large scale projects was that "applying the OO approach in the industrial setting turned out to be a battlefield strewn with landmines in a number of surprising areas". These landmines included different groups that had helped develop the project and therefore had different understandings of the problem, integration with other applications, and platform changes being part of the requirements. Also mentioned was data relationships that are complex and databases that are large, high performance requirements, solutions that have to be scaleable, a lot of inertia and resistance to change, and finally, lots of political issues.

The telephonic interview didn't reveal any relationship between company size and OO in South Africa. However, further interviews revealed that smaller companies would generally find it easier moving to OO, motivated by the need for a radical approach to be successful and competitive. Large companies typically have many products in other technologies in the market, as well as numerous legacy systems to maintain. This requires more co-ordination and communication. At the same time large companies are faced with more people resisting the change. The conclusion is that unlike small companies, in large companies a concept such as OO is not always pervasive and will not be accepted throughout.

Regarding the USA, Arranga used the notion of the one bad apple to explain why small companies are better for the adoption of OO. He justified this with the following reference from the legendary book The Mythical Man Month[5]:

"Cost does indeed vary as the product of the number of men and the number of months. Progress does not. ...Men and months are interchangeable commodities only when a task can be partitioned among many workers with no communication among them - this is true of reaping wheat or picking cotton, it is not even approximately true of systems programming"

The issue is intercommunication - if each part of the task must be separately co-ordinated with each other part, the effort increases as n(n-1)/2. Therefore adding more men lengthens instead of shortens the schedule.

He promoted this further asserting that in small companies people often wear different hats and not only do certain tasks, which can make the transition to a new technology such as OO much easier. Calabrese agreed that smaller companies would do better – in the Travelers Group project there were 30 developers.

The only vote against a smaller company size came from Dennis Laibson, arguing that OO requires a corporate decision and a large investment.

4.12.3 Project size and lifetime

The dilemma regarding project size was highlighted by Fred Brooks: *"This then is the problem with the small sharp team concept: it is too slow for really big systems" (if men and months traded evenly). ...For efficiency and conceptual integrity one prefers a few good minds doing design and construction. Yet for large systems one wants a way to bring considerable manpower to bear so that the product can make a timely appearance."* - [5]

Although the telephonic interview results did not reveal this, the popular opinion in South Africa was that smaller projects will do better in the transition to OO. Reasons given were having fewer people involved with differing opinions, better communication and less to change, even though in large companies OO comes to its true value because it scales well.

This was confirmed in the USA:

"Small empowered teams is key" - a comment made by a respondent in the 1995 COOUG survey.

The COOUG survey found that the average size of an OO design team was less than 5 people in most cases (38% of respondents) and 5 to 10 people in a further 37% of respondents. Calabrese reasoned that too big a group is bad, since daily reviews are important.

At Chrysler (UK), progress in development improved after the team was downsized from 35 to 16 developers.

Regarding project lifetime no significant conclusions could be drawn from the telephonic interviews. Also during the further interviews no consensus was reached. Arguments for longer projects doing better included that nothing sufficient can be ready in a year.

As mentioned before, many South African companies think they are implementing OO but are actually busy with rapid prototyping using Delphi etc. It is only when the projects get bigger that companies realise a solution cannot be hacked together. More time is required, together with a need to design ahead, which is why longer projects are doing better. Smaller projects do not need to be as formalised (it only needs to work). Long projects deal with more complexity, which is the real test.

The USA opinions were however unanimous; projects should be short. Calabrese mentioned that quite often with development, at first you cannot see much progress until it is all completed, which is why he suggests that a project should rather be short in life span.

Arranga and Price from Object-Z also suggested a small contained application that finishes quickly, again due to the disadvantages of having huge cycles.

4.13 Legacy systems

"While the advantages of using object-oriented design paradigm when embarking on the design of a new system have been established, the ramifications of using these techniques in the redesign of an existing system are less well-known"[40]

The significance of working with legacy systems can be found in Baer's comment: [12]

"The ability to identify successful strategies for working with these systems will determine the pace at which large organisations can move to object technology."

The local interviews revealed that most companies had systems they have classified as legacy systems. Interestingly, in quite a few cases these systems were relatively new (3 years), and some were even OO systems.[3] Methods of dealing with these legacy systems included:

- mapping the problem with a CASE tool,
- using messaging and CORBA for wrapping
- using queues to integrate where there is data entry in multiple places,

[3] Casais also referred to this problem in his article[7]

- file integration and data conversion (where data is the only contact between the old and new systems).

According to one USA company, legacy systems normally equate to being critical. The suggestions are therefore "leave as is" and only examining new functionality. Object-Z is currently working on strategies for using distributed object calls to the legacy systems. Wrappers were also mentioned as a possible solution.

4.14 The Role of Management

According to the Cutter Consortium's report[21] 41% of companies reported at least 1 failure in OO, one of the reasons given being poor management.

To find out whether there is any difference in the management of OO projects from other projects, one can refer to the comments made by the following authors:[23].

According to Kenny Rubin from ParcPlace,

"Although basic managerial goals remain unchanged when we use object technology, the specific ways in which we achieve the goals are changed. Object technology affects the software development processes, resources, and products. As such, a project lead by an experienced project manager, who does not have an understanding of object technology, is likely to fail due to ignorance surrounding required technical and organizational change. "

According to Fayad[15] the manager needs to deal with new or different problems: staffing, training, scheduling, cost estimation, standards, documentation, etc.

During the local interviews, although it seemed that management played an important role in the transition, there was no consensus as to whether the management of OO products is indeed different from the management of other projects. A consulting company found it to be so different that they in fact offer a course on the topic. The opinions expressed did not reflect a clear separation between the experienced and less experienced companies. Arguments given to support the change in management included the short turnaround when using an iterative approach, (in contrast with structured methods where there are long processes) that influences project planning, as well as the paradigm shift resulting in a new breed of programmers to manage.

In the USA, as in South Africa, there was no consensus on whether management is different for OO projects. According to Arranga, it requires a different mindset to manage OO projects due to different constraints. Managers need to know the technology and terminology. Management is a weak area and managers should therefore be mentored in this role just as developers need mentoring.

According to Price, management changes since a greater part of the time is now spent on analysis and design - this was also mentioned by Johnson from Rothwell International[23], saying that "OO technology changes the cost equations since coding becomes cheaper and more time is spent on design issues."

However, disagreement came from 2 sources:

Both Laibson (ObjectGems) and Calabrese (The Travelers Group) reasoned that management would be the same as before. Calabrese mentioned that if components are used, the system is divided into smaller tasks that are handled separately. They used traditional project management techniques with a new components-based architecture.

While it is therefore not clear whether the role of management has to change, the fact that management is important can again be emphasised looking at some of the comments made during the COOUG survey:

"Our management is ignorant of the benefits of OOT."

"OO is a long-term initiative that is tough to sell when you have immediate business requirements to satisfy."

4.15 Training

4.15.1 Why training is important

Another reason reported by the Cutter Consortium's report[21] for failure in OO, is the lack of experienced developers.

- file integration and data conversion (where data is the only contact between the old and new systems).

According to one USA company, legacy systems normally equate to being critical. The suggestions are therefore "leave as is" and only examining new functionality. Object-Z is currently working on strategies for using distributed object calls to the legacy systems. Wrappers were also mentioned as a possible solution.

4.14 The Role of Management

According to the Cutter Consortium's report[21] 41% of companies reported at least 1 failure in OO, one of the reasons given being poor management.

To find out whether there is any difference in the management of OO projects from other projects, one can refer to the comments made by the following authors:[23].

According to Kenny Rubin from ParcPlace,

"Although basic managerial goals remain unchanged when we use object technology, the specific ways in which we achieve the goals are changed. Object technology affects the software development processes, resources, and products. As such, a project lead by an experienced project manager, who does not have an understanding of object technology, is likely to fail due to ignorance surrounding required technical and organizational change. "

According to Fayad[15] the manager needs to deal with new or different problems: staffing, training, scheduling, cost estimation, standards, documentation, etc.

During the local interviews, although it seemed that management played an important role in the transition, there was no consensus as to whether the management of OO products is indeed different from the management of other projects. A consulting company found it to be so different that they in fact offer a course on the topic. The opinions expressed did not reflect a clear separation between the experienced and less experienced companies. Arguments given to support the change in management included the short turnaround when using an iterative approach, (in contrast with structured methods where there are long processes) that influences project planning, as well as the paradigm shift resulting in a new breed of programmers to manage.

In the USA, as in South Africa, there was no consensus on whether management is different for OO projects. According to Arranga, it requires a different mindset to manage OO projects due to different constraints. Managers need to know the technology and terminology. Management is a weak area and managers should therefore be mentored in this role just as developers need mentoring.

According to Price, management changes since a greater part of the time is now spent on analysis and design - this was also mentioned by Johnson from Rothwell International[23], saying that "OO technology changes the cost equations since coding becomes cheaper and more time is spent on design issues."

However, disagreement came from 2 sources:

Both Laibson (ObjectGems) and Calabrese (The Travelers Group) reasoned that management would be the same as before. Calabrese mentioned that if components are used, the system is divided into smaller tasks that are handled separately. They used traditional project management techniques with a new components-based architecture.

While it is therefore not clear whether the role of management has to change, the fact that management is important can again be emphasised looking at some of the comments made during the COOUG survey:

"Our management is ignorant of the benefits of OOT."

"OO is a long-term initiative that is tough to sell when you have immediate business requirements to satisfy."

4.15 Training

4.15.1 Why training is important

Another reason reported by the Cutter Consortium's report[21] for failure in OO, is the lack of experienced developers.

"What we learn from OO technology and its changes we need to apply to the other areas of business because it is clear that the business world will have to cope with increasing amount of change. Training is key" – Hazeltine.[24]

Page-Jones[40] found that successful object orientation requires more training in general software-engineering principles. He reasons that a structured system contains lines of code and modules which implies two levels of construction whereas an object-oriented system contains lines of code, methods and classes resulting in three levels of construction. Inheritance hierarchies also adds further complexity.

That training is a critical issue, is confirmed by Pancake[44]: Most dollars spent in converting to OT can be contributed to education and training. (There is the paradigm shift, as well as new programming language, software development techniques and tools to get familiar with)

And lastly, one can refer to Lato's[28] comment: *"Technologies come and go. In time the brand new technology of today will be a dinosaur. But a company that has learned how to learn will be able to adapt to the ever changing reality of doing business today."*

4.15.2 Is training happening?

The 1997 Cutter Consortium's report[21] concluded that most organisations invest heavily in training. Companies acquire the OO developers they need as follows: 51% train current staff in OO, 41% recruit staff with OO skills, 8% use consultants.

The January OMO survey[39] found that 42.1% of respondents had some form of classroom training in their primary methodology, whereas 57.1% had not.

According to the COOUG survey, 56% of responding companies had a commitment to retrain existing staff. At the Travelers Group, courses were undertaken in methodology, debugging classes, language (C++), walk throughs, and mentoring.

Results from the local telephonic interviews showed a clear relationship between the success in OO and the training level of developers: where developers had formal training, the adoption of OO was far more successful.

4.15.3 The problems

Pancake[44] found that there is a lack of good didactic examples based on real world needs, instead of the usual vending machine or ATM examples.

The problem noted during local interviews was finding people with practical OO experience.

This was confirmed in the USA: according to the Travelers Group in Connecticut, finding good OO people is the real challenge. At ObjectGems in Virginia only developers with 3-5 years experience are employed.

4.15.4 Guidelines

4.15.4.1 The order of training

Pancake[44] reasons that it is probably better to defer language training until the OO concepts are understood, since the learning hurdle is good design.

Bulman[11] recommends the order of training to be managers, analysts, designers, and then programmers. However he also mentions that the probability that many organisations will follow this path is very low.

During local interviews, the notion of "the 3 AHAs" was used to describe the learning process: first comes the language, then learning about objects, and lastly the paradigm shift and integration. It is this last phase that takes the longest and is the most important phase.

In the USA, Arranga, who teaches at UCLA, prescribes first teaching the language but not in too much detail. Ideally, OO should be taught in an iterative way just as OO itself is iterative. Laibson however recommended first teaching the concepts and then the language.

4.15.4.2 Mentoring

New developers should learn from the experience of others, without it being handholding or

outsourcing.[55] Manns[8] promotes a mentoring program to provide support, saying that in-house courses provide the prospect of educating some employees to become future trainers. Lato[28] also found that using mentoring is a good way of transferring knowledge from experienced people.

South African companies also favoured mentoring, where new recruits are assigned to older employees for guidance. This was also the practice in the USA.

4.15.4.3 On the use of consultants

Lato[28] found that investing in consulting enable companies to benefit from industry experience. It can provide an external check on the way work is done which can reduce the associated risk. Since many people will accept outside expertise more easily than in-house, buying outside experience often benefits the process.

In general, South African companies did not make use of the services of OO consulting companies - except where using individual contractors. The companies believe themselves to be more advanced and experienced than the level of service offered by the consulting companies. The high costs involved in employing consulting companies were also cited as a reason for not making use of such services.

Consulting companies often focus on certain market sectors and some companies felt that these consulting companies would not understand their culture. The comment "South Africa is driven too much by suppliers who think they are consultants" emphasised the need for product-independent consulting.

In contrast, in the USA Calabrese mentioned one of the mistakes made in the Travelers Group being that "they thought they could do without consultants but couldn't."

4.15.4.4 Who to train (the paradigm shift)

"The good procedural designers became the good object designers, probably because they had abstract reasoning capability"[30]

In South Africa a small software company felt that people who are good in C would also be good in C++. In contrast many companies mentioned that developers will have too many learning curves and that for OO one should rather use new people. A banking institution mentioned that it is easy for developers to use C++ or Smalltalk procedurally and that the only way to solve this is by getting new developers and using the old ones for maintenance.

Also in the USA there was a wide range of opinions on the issue. In the Travelers group, both existing and new staff were trained. As a result some developers were successful and others not - it seems to depend on the person's capabilities. Other sources however stated that you could not teach an old dog new tricks, new people do better.

Arranga cautioned against the danger of drawing a profile of the developer, saying that anybody with an open mind and right attitude can succeed. Similarly, a South African retail company found that the issue is attitude (towards OO) rather than aptitude, thereby supporting Vayda's argument[55] in choosing the right team.

4.15.4.5 Evolutionary vs. Revolutionary Approach

Dodani's article[14] takes a so-called shock therapy approach as a solution to the problem of immediately retraining highly qualified people in the workplace in OO.

South African companies were generally in favour of this revolutionary approach: EIKON mentioned that the waterfall approach does not work for OO and therefore the new process needs to be adopted together at once.

During the USA interviews, Arranga promoted the evolutionary approach but also mentioned the difficulty in applying it since people then tend to hold on to what they know.

One can therefore derive that the revolutionary approach is indeed the best way forward though evolutionary delivery is perhaps the best way of reducing project risks.

4.15.5 The state of training

Both locally and in the USA the opinion expressed numerously was that universities are still lagging behind industry in the teaching of OO (including the opinion from a lecturer at UCLA). This can however also be justified: The pace of change is just too fast to update the curriculum.

In most cases OO is merely a subtopic of software engineering. There is no overall course teaching all the elements of OO together.

Outside the academic arena, the situation does not look any better: courses are limited (the number of companies offering for example CORBA training is very small) and training is often coupled with product distribution where companies force the use of their product.

4.16 Language

The introduction of Object Orientation into programming languages can be summarised as given in Table 4.2:

Table 4.2 OO Languages

1969	Development of Smalltalk
1980	OO concepts are introduced into languages such as Pascal and Ada.
1985	Bjarne Stroustrup introduces C++ as a formal hybrid language. Steve Jobs develops the NeXT operating system under C++.
1987	Smalltalk is released for DOS, OS/2 Windows and Apple Mac Zortech releases C++ for DOS, UNIX, OS/2
1992	Borland is repositioned as a prime OO technology: (C++, Pascal etc.)

The 1997 Cutter Consortium's report[21] found that C++ and Java are the preferred OO languages while Microsoft Visual Studio is the most popular C++ environment.

The January 1998 survey in Object Magazine[39], found the primary programming language among respondents to C++ (49.3% of respondents), with Java being second (18.7%), Smalltalk third (16%) and Visual Basic getting 4% of the market.

The popularity of C++ has been justified by Reed[46]:"Because C++ is considered the successor to C, C programmers wanting the benefits of the object oriented paradigm look to C++ as a logical step toward bject oriented programming". Pancake disagrees[44] arguing that problems arise when people attempt to learn OT through a hybrid language. The author argues that C++ is the worst language to start with as it allows you to slip back into the habits you are used to, whereas languages such as Smalltalk and Self force you to think in an object oriented way.

Similarly, in South Africa, C++ dominated the market, with Visual Basic and Delphi competing for being the second most used. Java and Smalltalk shared third place.

Java was being chosen for platform independence and pure OO, while Smalltalk was a popular choice when searching for a pure and dynamic language that also offers garbage collection. As mentioned by Reed, C++ was often popular for historical reasons where a company moved there as a natural progression from C. It seemed as if the companies using Java, Smalltalk and C++ had more experience in OO while companies using, for example, Visual Basic were using it for its attractive tools, rather than its OO feat s. Perhaps this stems from the kind of development where more knowledge is necessary of the design principles itself and merely using a tool is not good enough.

According to a consulting company, the language should be the last thing to be chosen. First the architecture and a controlled development process should be in place.

The role that politics play in the choice of a language is evident from the comment during an interview at a merchant bank. "if with Smalltalk there are problems it's the language, if it is a C++ project, the problem is management or some other reason is given".

This sentiment was also shared in the USA interviews: In a small consulting company, the

original language choice was Smalltalk - due to Java's popularity they have since moved to Java, being more popular even though they felt that Smalltalk was technically superior.

The COOUG survey found that 36% of companies have standardised on C++ as language. This was also the case in the Travelers Group, where C++ was once again chosen because developers were already familiar with C.

4.17 Would an organisation attempting OO today make faster progress?

Fayad [30] argues that in the past, there were very few tools and methodologies available. However, a lot more became available to choose from which often lead to confusion about what to use. A report on the 1992 situation[11], revealed the numerous methodologies available at the time. The great diversity of methodologies was being advocated as a healthy sign as the technology was still developing.

Page-Jones [40] recently said the following: "... A shop currently embarking upon object orientation is no longer an early adopter and is no longer forced to navigate a pioneering voyage to Terra Incognita." One can therefore derive that the situation is currently more favourable for companies contemplating the transition to OO.

One of the reasons is the recent acceptance of the UML as a standard by the OMG (Object Management Group). Seldon[51] described the situation as follows: "By 1994 there were over 50 (OO) languages. As can be expected users could not find one language that catered for all their needs and this lead to the methods war, which saw various methodologies begin to incorporate each other's techniques."

In South Africa, no agreement could be reached about whether it would be easier to attempt OO now. Factors that could help are:

- There are formal methods and training available.
- OO is accepted now since the technology has been proven.
- more development tools are now available
- OO has not matured yet but has become practical to implement.
- It is best to start using OO now because the hype is over and the facts are on the table – especially for companies that are not leading edge companies.

Arguments against the statement:

- If there are existing skills from the old systems (structured methods) it is going to take longer.
- it will not be easier now because tools will not solve the problems - companies will make the same mistakes.

It was however also mentioned that the longer a company waits the longer it will typically sit with old technology which was paid for at a time when the company should have already moved to OO.

In the USA, there seemed to be more clarity on the issue: companies agreed that it would be easier to attempt OO now:

- the dust has settled, in standards and notation
- There are more resources available now. The COOUG survey reports that, in 1995, where companies have not progressed to OO yet, the reason given was a lack of experienced people.
- it is not a "fad" anymore (as referred to earlier by Page-Jones)
- due to standardisation, the number of techniques available has decreased.
- Being a more accepted technology now, a company is in a less visible position than earlier, which reduces risk.
- Debugging has taken place, which makes tools more fail proof.

4.18 Factors influencing speed of adoption

"3-5 years is necessary to change the way in which software is developed as well as acquiring an OO infrastructure." - the Cutter Consortium report. [21]

Looking at the factors that influence the speed of the adoption, there are numerous references to the

importance of management's role - it has to come from the top:

OO technology tends to come in through the back door, from the bottom up. The NASA Software Engineering Laboratory took 7 years to complete the transition to OO. Other organizations take less time because managers were committed to it. Litvintchouk's research concluded that what really determines the speed of insertion of OO is the approach to technology transition and to change management, more than the technical aspects. [30]

According to Orr,[44] "the problem is not the technology - it could be any technology.., the problem is cultural, organisational, its people focused"

This was confirmed during USA interviews. According to Arranga from Object-Z, the soft issues outweigh the technical issues, especially in the move from COBOL to OO. Factors mentioned influencing the speed of adoption are:

- Commitment from both the developers and management.
- Dedication.
- According to Calabrese (the Travelers Group) it is the success from other groups, making the technology more acceptable.
- Problems are often business related rather than technical, making the problem dollar driven.(Laibson – ObjectGems)

Locally, the following were mentioned as factors influencing the speed of adoption of OO in a company:

- Age of the developers: young people want to learn, older people are indoctrinated with structured principles and have prejudice. Together with this goes arguments for people fresh from university rather than people with years of COBOL experience.
- Knowledge of the process and the business.
- Commitment and willingness ("buy in" from the people) – this was mentioned many times.
- In contrast to the above argument it was mentioned that it is often the old people who have experienced the problems with the old technologies that are now willing to learn the new.
- In some companies unfortunately the "critical mass factor" is important – the company will not use a technology if the rest of the market does not.
- In one case it was felt that it is more a case of putting the (quality) process in place that made the difference rather than OO related issues.
- Technical skills.

4.19 Emerging technology

Both locally and in the USA there was agreement that Object Technology is still emerging.

The following reasons were given in the USA:

- Compilers are not mature yet.
- The technology will be emerging until more people have experience in the technology. (referring to OO COBOL specifically)

Locally the reasons given were:

- The standards are still evolving.
- "the technology is still changing rapidly and is not tried and trusted yet. There are risks. ". This comment often came from companies where OO was indeed still a new topic, which would explain the sentiment.
- According to a large insurance company the measure to use is the variety of tools still available.
- According to a merchant bank, areas where OO is still emerging includes component-based systems, distributed systems and databases. (The ODMG only published new specifications for databases recently). CORBA is also new. Therefore the next step for OO will be distributed object standards, standardised component software and pluggable items.

Graham[19] found that some people think that object technology is dead and that it will be replaced by "Component Technology".

The Cutter consortium report[21] concluded that most companies have started developing some form of component libraries, and 40% of these companies already have frameworks. Further evidence is a recent name change: Object Magazine became Component Strategies as a result of the evolution from object technology to this in vogue development.

A viewpoint matching the trend in literature, came from a South African consulting company, arguing that object technology was a thing of the early 1990s and that we have already transgressed into a component based phase.[4] This also relates to the opinion of a telecommunications company that the technology of the language is mature but that there is now growth in CORBA and the start of true components. The next step for OO will be distributed object standards, standardised component software and pluggable items. The San Francisco project[5] of IBM was also mentioned as the direction where OO is heading.

One of the sections of the Cutter Consortium report concentrates on how companies are using OO middleware. The results were as follows: the Internet (33%), CORBA (29%), DCOM (12%), Netware (8%), other (6%) and lastly none (12%).

Many companies are still investigating these standards. Both CORBA and COM are still rapidly evolving. Also, since the survey was done, there has been various new developments, including COM+, RMI as well as various products that will incorporate CORBA.[22]

The shift to components and standards for distributed objects that live on the web was also the agreed developments for the future according to most USA companies.

Further new developments in OO include the following:

- Next generation object oriented languages
- Security and safety critical software[47] - there is a growing interest in the use of formal methods for safety critical software.
- Server objects[52] which will make it possible to separate client design and server design. The important object languages (Smalltalk, C++) either currently support or have planned support for host class server environments. Clients are reaching the boundary of what they can provide without changing the server application. These workstation applications are becoming overly complex and will soon represent the next legacy problem.
- Reengineering OO legacy systems - according to Casais[7], companies that pioneered the move to OO now face the evolution of thousands of classes which represents a new kind of legacy systems. Casais commented that "given the pace at which all economic sectors are taking up OO, an OO reengineering technology is rapidly becoming an acute necessity." His article presents 18 approaches to reengineering OO systems.
- According to a large military company in South Africa next developments will be in databases and integration between information systems.
- According to a South African insurance house, OO has emerged in the last year but is not mature yet – the measure being the variety of tools still available. It is however exactly in this area (the tools) where most companies see new developments in the future, including fine tuning of the standard bodies and a shift in development tools where you will do less coding.

[4] This is also reinforced by the company's Web page motto: "Leaders in component based technology"

[5] The San Francisco Project is IBM's move towards providing reusability: it delivers on the promise of object-oriented programming by providing a set of server-based application frameworks - it consist of about 1000 object-oriented system-independent class libraries that provides developers with the necessary building blocks for developing server-based applications.[49]

5. Issues that need to be addressed

"One can expect the human race to continue attempting systems just within or just beyond our reach" - Brooks[5]

Various pitfalls in the transition to Object Orientation have been identified and guidelines provided. These lessons apply to any IT environment that relies on immature or fast changing technology. The following areas were identified where in general, locally or abroad, there seems to be a lack of research and solutions in the adoption of OO:

- Management: although the COOUG survey of 1995 reported that project management processes did not support OO projects at the time, there is not yet convincing proof that the situation is any better today. A comment made at one of the South African companies, is that "crisis management is the order of the day just to get things done"

- Legacy systems: As the name implies, legacy systems will be in existence for years to come. The means of handling these systems need to be refined.

- From legacy systems to people: Although with time the need for retraining procedurally thinking developers should disappear as new, object-thinking people enter the marketplace, a gap currently exists for successfully training and retraining existing developers to overcome the paradigm shift.

- Metrics and CASE - both have seen very little use, while the opportunities for improving the way in which OO systems are developed, are unlimited.

Being problematic in the adoption of OO, these issues will also influence the adoption of future new technologies.

6. Conclusions

This article served to describe the various topics involved in the adoption of Object Orientation.

Going back to the perceived differences, the statement regarding "islands of excellence" is most appropriate to describe the comparison between South African and USA companies. Although not on the same scale, the progress towards OO in South Africa is largely similar and in some ways ahead of the adoption in the USA.

South African companies have perhaps not paid as much attention to the change in the organisational structure that is required, and are sometimes choosing Object Orientation for the wrong reasons. Technically, however, most companies have achieved their goal in adopting OO, without the presence of a Fowler or a Booch to lend a helping hand.

7. Reference

[1] Anderson B., Shaw M., Best L. and Beck K. Software Architecture: The Next Step for Object Technology, *OOPSLA '93 Proceedings*, Sept. 1993, pp. 356-359. and Panel Session pp. 63-66.

[2] Basili V. Briand L.C. and Melo W.L. How reuse influences productivity in object-oriented systems, *Communications of the ACM*, vol. 39, no. 10, Oct. 1996, pp. 104-116.

[3] Benner K. Position Paper for OOPSLA Workshop: Are Object-oriented CASE Frameworks Ready for Prime Time, http://www.compsvcs.com/cs3/benner.html

[4] Berg W. Cline M. and Girou M. Lessons Learnt from the OS/400 OO Project, *Communications of the ACM*, vol. 38, no. 10, Oct. 1995, pp. 54-64.

[5] Brooks F. The Mythical Man-Month: Essays on Software Engineering, 1982, pp. 16-177.

[6] Calabrese R. The COOUG 1995 Survey results, http://www.cooug.org/b/newslets/survey.htm

[7] Casais E. Re-Engineering Object-Oriented Legacy Systems, *Journal of Object-Oriented Programming*, Jan. 1998, pp 45-52.

[8] Chidamber S.R. and Kemerer C.F. Towards a metrics suite for object oriented design, *OOPSLA '91 Proceedings*, Oct. 1991, pp. 197-211.

[9] Cohen J. Manns M. Lilly S. Gabriel R.P. Conway J. and D'Souza D. PANEL: Training Professionals In Object Technology, *OOPSLA '94 Proceedings*, Oct. 1994, pp. 46-50.

[10]Corporate and Professional Publishing Group, *Refactoring*, http://www2.awl.com/cseng/titles/0-201-89542-0/Refactoring.htm

[11]De Champeaux D. and Faure P. A comparative study of object-oriented analysis methods, *Journal of Object Oriented Programming*, Mar/Apr. 1992, pp. 21-32.

[12]De Champeaux D. Balzer B. Bulman D. Culver-Lozo K. Jacobson I. And Mellor S.J. The OO Software Development Process, *OOPSLA '92 Proceedings*, Oct. 1992, pp. 484-489.

[13]De Champeaux D. Baer A.J. Bernsen B. Korncoff A.R. Korson T. and Tkach D.S. Strategies for Object-Oriented Technology Transfer, *OOPSLA '93 Proceedings*, Oct. 1993, pp. 437-447.

[14]Dodani M. Object-oriented shock therapy, *Journal of Object Oriented Programming*, Jul/Aug. 1996, pp. 17-19.

[15]Fayad M.E. and Tsai W. Object-Oriented Experiences, *Communications of the ACM*, vol. 38, no. 10, Oct. 1995, pp.51-53

[16]Fayad M.E. Tsai W. and Fulghum M.L. Transition to Object-Oriented Software Development, *Communications of the ACM*, vol.39, no.2, Feb. 1996, pp.108-121.

[17]Flensted-Jensen N. Position Paper for OOPSLA Workshop: Are Object-Oriented CASE Frameworks Ready for Prime Time, http://www.compsvcs.com/cs3/neils.html

[18]Fraser S. Beck K. Booch G. Coleman D. Coplien J. Helm R. and Rubin K. How Do Teams Shape Objects – How Do Objects Shape Teams?, 1994, pp. 468-473. and Panel Session pp. 63-66.

[19]Graham I. In Search of the Three P.st Books *Journal of Object Oriented Programming*, Sept. 1997, pp. 43-45.

[20]Graham I. Migrating to object technology, Addison-Wesley UK, 1995, pp.3-72, pp.399-444.

[21] Harmon P. The corporate use of object technology, Cutter Consortium, http://www.cutter.com/itgroup/reports/corpuse.html 1997.

[22]Harmon P. Distributed Computing Architecture, Cutter Consortium, http://www.cutter.com/consortium/architecture/corbacom.html, 1998.

[23]Hill L. Rubin K. Daniels J. Berman C. Coplien J. and Johnson D. Managing Object Oriented Projects, *OOPSLA '95 Proceedings*, Oct. 1995, pp.88-90.

[24] Korson T. The Role Of A Corporate Object Technology Center, *OOPSLA '93 Proceedings*, Oct. 1993, pp.131-134.

[25]Korson T, Managing the Transition to Object-Oriented Technology, *OOPSLA '91 Proceedings*, Oct. 1991, pp. 55-62.

[26]Kristek T. and Vaishnavi V. Role of a Corporate Technology Center, *OOPSLA '94 Proceedings*, Oct. 1994, pp. 72-77.

[27]LaBoda D.M. and Ross J.W. CASE study, Travelers Property Casualty Corporation: Building an Object Environment for Greater Competitiveness, *Centre for Information Systems Research*, Massachusetts Institute of Technology, Sept.1997, pp.1-20.

[28]Lato K. Learn to learn: Training on new technology, *Journal of Object Oriented Programming*, Mar/Apr. 1997, pp. 24-27.

[29]Lee W. How To Adapt OO Development Methods In A Software Development Organisation – A Case Study, *OOPSLA '9 Proceedings*, Oct. 1994, pp. 19-24.

[30]Litvintchouk S.D. Evolving Towards Object-Oriented Technology In Large Organizations, *OOPSLA '93 Proceedings*, Oct. 1993, pp. 73-76.

[31]Malan R. Coleman D. and Letsinger R. Lessons from the Experiences of Leading Edge

Object Technology Projects in Hewlett-Packard, *OOPSLA '95 Proceedings*, Oct. 1995, pp. 33-46.

[32]Manns M.L. and Nelson H.J. Retraining procedure-oriented developers: An issue of skill transfer, *Journal of Object Oriented Programming*, Nov/Dec. 1996, pp. 6-10.

[33]Meszaros G. Experience Building Large OO Frameworks at BNR, *OOPSLA '94 Proceedings*, Oct. 1994, pp. 14-18.

[34]Meyer B. Towards an O-O Curriculum, http://www.tools.com/doc/manuals/technology/curriculum/index.html

[35]Moreau D.R. and Dominick W.D. Object-Oriented Graphical Information Systems: Research Plan and Evaluation Metrics, *The Journal of Systems and Software*, vol.10, 1989, pp.23-28.

[36]Narayanaswamy K. Are Object Oriented CASE Frameworks Ready for Prime Time, *OOPSLA '95 Proceedings*, Oct. 1995, pp. 155-158.

[37]Noble J. and Kolbe K Position Paper: Are Object Oriented CASE Frameworks Ready for Prime Time, http://www.compsvcs.com/cs3/noble.html

[38]Object Magazine Online, The Object Magazine Online Interview -Bjarne Stroustrup, 1998, http://www.objectmagazine.com

[39]The Object Magazine Online Survey: What are you using?, Jan. 1998, http://www.sigs.com/omo/questionnaire/results.html

[40]Olander G. Experience Report - Chembench: Redesign of a Large Commercial Application Using Object-Oriented Techniques, *OOPSLA '92 Proceedings*, Oct. 1992, pp. 13-16.

[41]Ovum Evaluates: CASE Products, Why size is important, Oct. 1995, http://www.ovum.com/evaluate/case/index.html

[42]Page-Jones M. Object Orientation: Making the Transition, copyright Wayland Systems Inc. 1997, pp. 1-19.

[43]Page-Jones M. The seven stages in software engineering, *American Programmer*, Jul/Aug. 1990.

[44]Pancake C.M. The Promise and the Cost of Object Technology: A five-year forecast, *Communications of the ACM*, Oct. 1995, vol.38, no.10, pp.33-49.

[45]Pickering C. Survey of advanced technology, copyright Systems Development Inc., 1993.

[46]Reed D.R. Cagan M. Goldstein T. and Moo B. Issues in Moving from C to C++, *OOPSLA '91 Proceedings*, Oct. 1991, pp. 163-165.

[47]Riehle R. Reuse OOP and Safety-Critical Software, *Journal of Object Oriented Programming*, Nov/Dec. 1997, pp. 21-24.

[48]Rivas E.A. Position Paper for OOPSLA Workshop: Are Object-Oriented CASE Frameworks Ready for Prime Time, http://www.compsvcs.com/cs3/rivas.html

[49]Scannell E. IBM rolls out building blocks for San Francisco Project apps, Infoworld, vol. 18, issue 52/53, Dec. 1996, http://www.ibm.com/Java/Sanfrancisco/

[50]Schach SR. Classical and Object-oriented software engineering, third edition, Irwin/McGraw-Hill, USA 1996, pp. 70-74.

[51]Seldon A. The basics of object standards, *Computerweek*, Aug. 1997 p. 16.

[52]Shan Y-P. Morgan T. Proudfoot P. Thompson J. Tibbetts J. and Woolfrey A. Objects on the Server: Are We Ready, *OOPSLA '96 Proceedings*, Oct. 1996, pp. 384-388.

[53]Shan Y-P. Auer K. Bear A.J. Adamczyk J. Goldberg A. Love T. and Thomas D. Smalltalk in the Business World the Good the Bad and the Future, *OOPSLA '94 Proceedings*, Oct. 1994, pp. 145-152.

[54]Standish Group International, The Standish Group Survey Says: Object-Oriented Programming is a must, http://prolifics.com/anan/standish2.html, 1997.

[55]Vayda T.P. Lessons From the Battlefield, *OOPSLA '95 Proceedings*, Oct. 1995, pp. 439-452.

[56]Wick M.R. On using C++ and Object-Orientation in CS1: The message is still more important than the medium, *SIGCSE '95*, Mar. 1995, pp. 322-326.

[57]Yourdon E. Object-oriented analysis and design seminar, 1994, pp. 1-150.

8. Author Contact Details

Contact Details: **Miranda Jansen van Rensburg,** Technikon SA (Information Technology), C de Wet Drive Roodepoort Office Phone: +27-11-471-2929, Fax: +27-11-471-3270, Internet E-mail: mjvanren@tsamail.trsa.ac.za

■

2 Discussion and conclusions

Object Technology has come a long way from being the "superfad" that Page-Jones described it to be[76]. Numerous successful experiences have been reported. Various statistics are available describing the momentum that Object Orientation (OO) has gained around the world[3].

The principles of OO were around for more than two decades before it was widely accepted. It is however not only leading edge companies' territory anymore. Numerous companies that only support technologies used by the critical mass, now invest in OO.

The view that this progress was not without difficulty instigated this research. The aim was to find out what these obstacles are and how successful companies have dealt with each of these obstacles.

By studying the progress in companies that have had success and companies that have had failure, an unbiased recommendation could be formulated for handling these obstacles in future.

Upon completion of the literature study, an abundance of information and experiences regarding companies abroad was available. Experiences of successful OO implementations more than ten years ago have been reported.

In contrast, the situation in South Africa was not documented clearly. During an interview, the reason given as explanation for this was that "the people doing little talk a lot, people who do the work stay quiet". In other words, companies who were making a success of their OO adoption considered it to be a competitive advantage and were therefore not keen to write about it.

Through a series of interviews, including short telephonic interviews and longer, in-depth face-to-face discussions, the current situation regarding the implementation of OO was exposed. Surprisingly, companies were very honest about their progress as well as their mistakes and problems.

A clear segregation of issues was exposed that are involved in the transition to OO. Some of these are:

- The reasons why companies move to OO range from the traditional reasons such as maintainability (especially in the USA) to more unconventional reasons such as satisfying developers' needs. Due to

skill shortages, South African companies specifically experienced this phenomenon.

- The promise of reuse: many companies have not yet achieved reuse but are also not perturbed by this, since in most cases it was not the reason for adopting OO originally.

- Factors influencing the speed of the adoption: the importance of the "soft" issues more than the pure technical issues. This became evident in the type of problems experienced as well as the forces behind decisions such as language choice etc.

- The profile of the company

- The importance of the organisational structure

Once the issues were uncovered, it was possible to compare the experiences of South African companies with their overseas counterparts. Similar interviews were conducted in the USA, to find out if South African companies are indeed lagging behind the international trends as they presume to be.

In many of the issues, the results were similar. The difference was smaller than most South African companies expected it to be. As was found in the USA, there are also islands of excellence in South Africa, only fewer. The fact that South African companies have advanced this far without the presence of numerous OO gurus or OO-related conferences, and without the ability to purchase expensive tools, indicates even more progress. As a result of a major skills shortage, South African companies are often forced to only support major technologies, revealing even more success on the local front.

Turning back to the global situation, areas of concern were identified, where theoreticians' views have been ignored, and both South African and overseas companies have not implemented any of the suggestions made, including:

- management: no consensus exists on whether this role has changed

- training

- the need for a quality process

- CASE tools and metrics

- Integration of legacy systems

Looking at some recent name changes in this field[1], it is uncertain whether OO will continue to exist in its current format. Many companies see it evolving into a component based technology.

Being one of the most difficult changes experienced so far in the relatively new world of software development, many of the lessons learnt in the transition will certainly apply in the challenges that lie ahead in the forever changing IT world.

It is therefore suggested that the lessons learnt within the technical context of OO development, will continue to be applicable:

Training should become iterative just as the software development process itself should be iterative.

Leading edge companies who have discovered the means of training, managing and retaining skilled people will continue to profit from the lessons they have learnt.

Just as the development of the architecture has been stressed and developed in this transition, the importance of organising the people has become evident. The changing organisational structure will continue to be important.

Finally, just as the principle of reuse has been stressed, the lessons learnt by companies that have successfully overcome the obstacles discussed, will be reused in the technology transitions to come.

■

[1] Object Magazine became Component Strategies; a South African consulting company specialising in OO changed its motto to be component specific.

2 Project Bibliography and References

[1] Alabiso B. Transformation of data flow analysis model to object oriented design, *OOPSLA '88 Proceedings*, Sept. 1988, pp. 335-353.

[2] Aksit M. and Bergmans L. Obstacles in Object-Oriented Software Development, *OOPSLA '92 Proceedings*, Oct. 1992, pp. 341-358.

[3] Anderson B., Shaw M., Best L. and Beck K. Software Architecture: The Next Step for Object Technology, *OOPSLA '93 Proceedings*, Sept. 1993, pp. 356-359. and Panel Session pp. 63-66.

[4] Arnold T.R. and Fuson W.A. Testing "In a Perfect World", *Communications of the ACM*, vol. 37, no. 9, Sept. 1994, pp. 78-86.

[5] Artsy Y.S. Position Paper for OOPSLA Workshop: Are Object-Oriented CASE Frameworks Ready for Prime Time, http://www.compsvcs.com/cs3/artsy.html

[6] Baker M. Position Paper for OOPSLA Workshop: Are Object-Oriented CASE Tools ready for Prime Time, http://www.compsvcs.com/cs3/baker.html

[7] Barclay P.J. and Jackson. S.J. Object-Oriented Programming Transition Strategies, *OOPSLA '93 Proceedings*, Sept. 1993, pp. 39-40.

[8] Barton D. and Arnold T. Evolving to Objects - The Witches' Brew, *OOPSLA '95 Proceedings*, Oct. 1995, pp. 414-425.

[9] Basili V. Briand L.C. and Melo W.L. How reuse influences productivity in object-oriented systems, *Communications of the ACM*, vol. 39, no. 10, Oct. 1996, pp. 104-116.

[10]Beck K. A Laboratory for Teaching Object-Oriented Thinking, *OOPSLA '89 Proceedings*, Oct.1989, pp. 1-6.

[11]Benner K. Position Paper for OOPSLA Workshop: Are Object-oriented CASE Frameworks Ready for Prime Time, http://www.compsvcs.com/cs3/benner.html

[12]Berg W. Cline M. and Girou M. Lessons Learnt from the OS/400 OO Project, *Communications of the ACM*, vol. 38, no. 10, Oct. 1995, pp. 54-64.

[13]Brooks F. The Mythical Man-Month: Essays on Software Engineering, Addison-Wesley, 1982, pp. 16-177.

[14]Calabrese R. The COOUG 1995 Survey results, http://www.cooug.org/b/newslets/survey.htm

[15]Casais E. Re-Engineering Object-Oriented Legacy Systems, *Journal of Object-Oriented Programming*, Jan. 1998, pp. 45-52.

[16]Chambers C. and Ungar D. Making Pure Object-Oriented Languages Practical, *OOPSLA '91 Proceedings*, Oct. 1991, pp. 1-15.

[17]Chidamber S.R. and Kemerer C.F. Towards a metrics suite for object oriented design, *OOPSLA '91 Proceedings*, Oct. 1991, pp. 197-211.

[18]Clyde S.W. Embley D.W. and Woodfield S.N. Tunable formalism in object-oriented systems analysis: meeting the needs of both theoreticians and practitioners, *OOPSLA '92 Proceedings*, Oct. 1992, pp. 452-465.

[19]Coad P. Finding Objects: Practical Approaches, *OOPSLA '91 Proceedings*, Oct. 1991, pp.17-19.

[20]Cohen J. Manns M. Lilly S. Gabriel R.P. Conway J. and D'Souza D. PANEL: Training Professionals In Object Technology, *OOPSLA '94 Proceeding.*, Oct. 1994, pp. 46-50.

[21]Corporate and Professional Publishing Group, *Refactoring*, http://www2.awl.com/cseng/titles/0-201-89542-0/Refactoring.htm

[22]Cummins F. Cunis R. and Lamping J. Next Generation Object-Oriented Programming Languages: Extending the Paradigm, *OOPSLA '93 Proceedings*, Sept. 1993, pp. 91-93.

[23]De Champeaux D. Anderson A. and Feldhousen E. Case Study of Object-Oriented Software Development, *OOPSLA '92 Proceedings*, Oct. 1992, pp. 377-391.

[24]De Champeaux D. and Faure P. A comparative study of object-oriented analysis methods, *Journal of Object Oriented Programming*, Mar/Apr. 1992, pp. 21-32.

[25]De Champeaux D. Balzer B. Bulman D. Culver-Lozo K. Jacobson I. And Mellor S.J. The OO Software Development Process, *OOPSLA '92 Proceedings*, Oct. 1992, pp. 484-489.

[26]De Champeaux D. Baer A.J. Bernsen B. Korncoff A.R. Korson T. and Tkach D.S. Strategies for Object-Oriented Technology Transfer, *OOPSLA '93 Proceedings*, Oct. 1993, pp. 437-447.

[27]De Champeaux D. Constantine L. Jacobson I. Mellor S. Ward P. and Yourdon E. Structured Analysis and Object Oriented Analysis, *ECOOP/OOPSLA '90 Proceedings*, Oct. 1990, pp. 135-139.

[28]DeNatale R. The Role of Methods and Case in OO Development, *OOPSLA '92 Proceedings*, Oct. 1992, pp. 39-47.

[29]Dietrich W.C. Nackman L.R. and Gracer F. Saving a Legacy with Objects, *OOPSLA '89 Proceedings*, Oct. 1989, pp. 77-83.

[30]Dodani M. Object-oriented shock therapy, *Journal of Object Oriented Programming*, Jul/Aug. 1996, pp. 17-19.

[31]Fayad M.E. and Tsai W. Object-Oriented Experiences, *Communications of the ACM*, vol. 38, no. 10, Oct. 1995, pp.51-53

[32]Fayad M.E. Tsai W. and Fulghum M.L. Transition to Object-Oriented Software Development, *Communications of the ACM*, vol.39, no.2, Feb. 1996, pp.108-121.

[33]Flensted-Jensen N. Position Paper for OOPSLA Workshop: Are Object-Oriented CASE Frameworks Ready for Prime Time, http://www.compsvcs.com/cs3/neils.html

[34]Fraser S. Beck K. Booch G. Coleman D. Coplien J. Helm R. and Rubin K. How Do Teams Shape Objects – How Do Objects Shape Teams?, *OOPSLA '94 Proceedings*, Oct. 1994, pp. 468-473 and Panel Session pp. 63-66.

[35]Fraser S. Marshall L. and Bailetti T. Workshop Report – Team Approaches to OO Design, *OOPSLA '92 Proceedings*, Oct. 1992, pp. 85-92.

[36]Graham I. In Search of the Three Best Books, *Journal of Object Oriented Programming*, Sept 1997, pp. 43-45.

[37]Graham I. Migrating to object technology, Addison-Wesley UK, 1995, pp. 3-72, pp. 399-444.

[38]Harmon P. The corporate use of object technology, Cutter Consortium, http://www.cutter.com/itgroup/reports/corpuse.html 1997.

[39]Harmon P. Distributed Computing Architecture, Cutter Consortium, http://www.cutter.com/consortium/architecture/corbacom.html, 1998.

[40]Harrison W.H. Shilling J.J. and Sweeney P.F. Good News, Bad News: Experience Building A Software Development Environment Using The Object-Oriented Paradigm, *OOPSLA '89 Proceedings*, Oct. 1989, pp. 85-94.

[41]Hartman M. Jewell F.W. Scott C. and Thornton D. Taking An Object-Oriented Methodology Into The Real World, *OOPSLA '94 Proceedings*, Oct. 1994, pp. 25-30.

[42]Herndon W.K. Sandhu R. and Demurjian S. The Standards Are Coming! Standards For Security In Object-Oriented Systems, *OOPSLA '94 Proceedings*, Oct. 1994, pp. 92-95.

[43]Hill L. Rubin K. Daniels J. Berman C. Coplien J. and Johnson D. Managing Object Oriented Projects, *OOPSLA '95 Proceedings*, Oct. 1995, pp.88-90.

[44]Hφydalsvik G.M. and Sindre G. On The Purpose Of Object-Oriented Analysis, *OOPSLA '93 Proceedings*, Oct. 1993, pp. 240-255.

[45]Jacobson I. and Lindstrom F. Re-engineering of old systems to an object-oriented architecture, *OOPSLA '91 Proceedings*, Oct. 1991, pp. 340-350.

[46]Kerth N.L. A Structured Approach To Object-Oriented Design, *OOPSLA '91 Proceedings*, Oct. 1991, pp. 21-43.

[47]Korson T. Managing the Transition to Object-Oriented Technology, *OOPSLA '91 Proceedings*, Oct. 1991, pp. 55-62.

[48]Korson T. The Role Of A Corporate Object Technology Center, *OOPSLA '93 Proceedings*, Oct. 1993, pp.131-134.

[49]Kristek T. and Vaishnavi V. Role of a Corporate Technology Center, *OOPSLA '94 Proceedings*, Oct. 1994, pp. 72-77.

[50]LaBoda D.M. and Ross J.W. CASE study, Travelers Property Casualty Corporation: Building an Object Environment for Greater Competitiveness, *Centre for Information Systems Research*, Massachusetts Institute of Technology, Sept. 1997, pp. 1-20.

[51]Laffra C. and Malhotra A. Advanced Techniques For Understanding Profiling And Debugging Object Oriented Systems, *OOPSLA '93 Proceedings*, Oct. 1993, pp. 79-81.

[52]Lato K. Learn to learn: Training on new technology, *Journal of Object Oriented Programming*, Mar/Apr. 1997, pp. 24-27.

[53]Laurance D.C. and Lewis N. Position Paper for OOPSLA Workshop: Are Object-Oriented CASE Frameworks Ready For Prime Time, http://ww .v.compsycs.com/cs3/att.html

[54]Lee W. How To Adapt OO Development Methods In A Software Development Organisation – A Case Study, *OOPSLA '94 Proceedings*, Oct. 1994, pp. 19-24.

[55]Li W. Henry S. Kafura D. and Schulman R. Measuring object-oriented design, *Journal of Object-Oriented Programming*, Jul/Aug. 1995, pp. 48-55.

[56]Litvintchouk S.D. Evolving Towards Object-Oriented Technology In Large Organizations, *OOPSLA '93 Proceedings*, Oct. 1993, pp. 75-76.

[57]Liu C. Goetze S. and G ynn B. What Contri tes To Successful Object-Oriented Learning, *OOPSLA '92 Proceedings*, Oct. 1992 pp. 77-86.

[58]Love T. Avoiding Uninteresting Old Mistakes, *Journal of Object Oriented Programming*, Jul/Aug 1997, pp. 14-1

[59]Malan R. Coleman D. and Letsinger R. Lessons from the Experiences of Leading Edge Object Technology Projects in Hewlett-Packard, *OOPSLA '95 Proceedings*, Oct. 1995, pp. 33-46.

[60]Malloy M.A. Post-Mortem Assessment of Interface Changes for an Evolving Object- Oriented, "Not-So-Rapid" Prototype, *OOPSLA '93 Proceedings*, Oct. 1993, pp. 17-18.

[61]Manns M.L. and Nelson H.J. Retraining procedure-oriented developers: An issue of skill transfer, *Journal of Object Oriented Programming*, Nov/Dec. 1996, pp. 6-10.

[62]Martin R. and Ottinger T. Limitations of OO? Usenet Discussion Commentary, *Object Magazine Online*, http://www.objectmagazine.com/tips/9711/martin.html

[63]Martin R. and Ottinger T. Limitations of OO? Usenet Discussion Commentary Part 2, *Object Magazine Online*, http://www.sigs.com/omo/tips/martin.html

[64]Meszaros G. Experience Building Large OO Frameworks at BNR, *OOPSLA '94 Proceedings*, Oct. 1994, pp. 14-18.

[65]Meyer B. Towards an O-O Curriculum, http://www.tools.com/doc/manuals/technology/curriculum/index.html.

[66]Meyers G.L. Position Paper for OOPSLA Workshop: Are Object-Oriented CASE Frameworks Ready for Prime Time, http://www.compsvcs.com/cs3/meyers.html

[67]Moreau D.R. and Dominick W.D. Object-Oriented Graphical Information Systems: Research Plan and Evaluation Metrics, *The Journal of Systems and Software*, vol.10, 1989, pp.23-28.

[68]Narayanaswamy K. Are Object Oriented CASE Frameworks Ready for Prime Time, *OOPSLA '95 Proceedings*, Oct. 1995, pp. 155-158.

[69]Noble J. and Kolbe K. Position Paper: Are Object Oriented CASE Frameworks Ready for Prime Time, http://www.compsvcs.com/cs3/noble.html

[70]Object Magazine Online, The Object Magazine Online Interview - James Coplien, 1998, http://www.objectmagazine.com

[71]Object Magazine Online, The Object Magazine Online Interview - Bjarne Stroustrup, 1998, http://www.objectmagazine.com

[72]The Object Magazine Online Survey: What are you using?, Jan. 1998, http://www.sigs.com/omo/questionnaire/results.html

[73]The Object Magazine Online Survey: What's Your Biggest Concern?, Jan. 1998, http://www.objectmagazine.com/questionnaire/9801/results.html

[74]Olander G. Experience Report - Chembench: Redesign of a Large Commercial Application Using Object-Oriented Techniques, *OOPSLA '92 Proceedings*, Oct. 1992, pp. 13-16.

[75]Ovum Evaluates: CASE Products, Why size is important, Oct. 1995, http://www.ovum.com/evaluate/case/index.html

[76]Page-Jones M. Object Orientation: Making the Transition, copyright Wayland Systems Inc. 1997, pp. 1-19.

[77]Page-Jones M. The seven stages in software engineering, *American Programmer*, Jul/Aug. 1990.

[78]Pancake C.M. The Promise and the Cost of Object Technology: A five-year forecast, *Communications of the ACM*, Oct. 1995, vol 38, no.10, pp.33-49.

[79]Pickering C. Survey of advanced technology, copyright Systems Development Inc, 1993.

[80]Pircher P. and Coggins M. Position Paper for OOPSLA Workshop: Are Object-Oriented CASE Frameworks Ready for Prime Time, http://www.compsvcs.com/cs3/pircher.html

[81]Reed D.R. Cagan M. Goldstein T. and Moo B. Issues in Moving from C to C++, *OOPSLA '91 Proceedings*, Oct. 1991, pp. 163-165.

[82]Riehle R. Reuse OOP and Safety-Critical Software, *Journal of Object Oriented Programming*, Nov/Dec. 1997, pp. 21-24.

[83]Rivas E.A. Position Paper for OOPSLA Workshop: Are Object-Oriented CASE Frameworks Ready for Prime Time, http://www.compsvcs.com/cs3/rivas.html

[84]Rosson M. and Carroll J.M. Scaffolded examples for learning object-oriented design, *Communications of the ACM*, vol. 39, no. 4, Apr. 1996, pp. 46-47.

[85]Scannell E. IBM rolls out building blocks for San Francisco Project apps, Infoworld, vol. 18, issue 52/53, Dec. 1996, http://www.ibm.com/Java/Sanfrancisco/

[86]Schach SR. Classical and Object-oriented software engineering, third edition, Irwin/McGraw-Hill, USA 1996, pp. 70-74.

[87]Schedlbauer M. How to Select a Training Organisation, *Journal of Object Oriented Programming*, Oct. 1997, pp. 39-40.

[88]Seldon A. The basics of object standards, *Computerweek*, Aug. 1997 p. 16.

[89]Shan Y-P. Morgan T. Proudfoot P. Thompson J. Tibbetts J. and Woolfrey A. Objects on the Server: Are We Ready, *OOPSLA '96 Proceedings*, Oct. 1996, pp. 384-388.

[90]Shan Y-P. Auer K. Bear A.J. Adamczyk J. Goldberg A. Love T. and Thomas D. Smalltalk in the Business World the Good the Bad and the Future, *OOPSLA '94 Proceedings*, Oct. 1994, pp. 145-152.

[91]Shih T.K. Wang C-C. and Chung C-M. Using Z to specify object-oriented software complexity measures, *Information and Software Technology*, vol.39, 1997, pp. 515-529.

[92]Standish Group International, The Standish Group Survey Says: Object-Oriented Programming is a mus., http://prolifics.com/anan/standish2.html, 1997.

[93]Stark M. Impacts of object-oriented technologies: seven years of SEL studies, *OOPSLA '93 Proceedings,* Oct. 1993, pp. 365-373.

[94]Townsend P. and Murphy G. Experience Report – Objects in the Life-Cycle, *OOPSLA '92 Proceedings,* Oct. 1992, pp. 25-28.

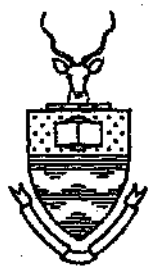
[95]Vaishnavi V. and Korson T. Role of a Corporate Object Technology Center, *OOPSLA '95 Proceedings,* Oct. 1995, pp.186-190.

[96]Vayda T.P. Lessons From the Battlefield, *OOPSLA '95 Proceedings,* Oct. 1995, pp. 439-452.

[97]Viljoen P. Object Orientation: Concepts and Method, version 4, copyright Infomet, 1992, pp.1-123.

[98]Wick M.R. On using C++ and Object-Orientation in CS1: The message is still more important than the medium, *SIGCSE '95,* Mar. 1995, pp. 322-326.

[99]Yourdon E. Object-oriented analysis and design seminar, 1994, pp. 1-150.



TRANSITION IN METHODOLOGIES

Master Document List

Management Product

Version 1.01

Document Status: Approved

Table of Contents

Table of Contents ii

Change History iii

Configuration Control iii

Document History iii

Revision History iii

Management Authorisation iii

Change Forecast iii

1 Scope 1

1.1 Introduction 1

1.2 Purpose 1

1.3 Applicability 1

1.4 Definitions 1

1.5 Audience 1

1.6 Applicable Documents 2

1.7 Assumptions 2

1.8 Requirements Traceability 2

1.9 Procedures 2

2 Master Document List 6

Change History

Configuration Control

Project:	TRANSITION IN METHODOLOGIES
Title:	Master Document List
Doc. Reference:	C:\MSC\MP\NDM001.010
Created by:	M Van Rensburg
Creation Date:	12 February 1998

Document History

Version	Date	Status	Who	Saved as:
0.10	98/02/12	Draft	MVR	NDM001.010
1.00	98/04/22	Approved	MVR	NDM001.100

Revision History

Version	Date	Changes
0.10	98/02/12	New document created using QST00110.106
1.00	98/04/22	Document approved by project supervisor
1.00	98\06\17	Added NDM901 as well as several documents containing minutes and agendas to the MDL
1.01	98\06\22	Prof Walker corrected the footers.
1.01	98\09\23	Added new documents including web page and minutes

Management Authorisation

Version	Date	Status	Management Board Minute Reference
1.00	98/04/22	Approved	98\04\22 2.4

Change Forecast

This document will be updated each time a new documented element is added into the Transitions in Methodologies Project.

1 Scope

1.1 Introduction

The research done in this project looks at areas where change was undertaken from a previously structured design to an OO design and also where informal or no design principles were previously used.

The research report will provide guidelines for companies who are currently contemplating a change to the OO methodology, covering important issues one should know about prior to this change.

It will also summarise the problems faced in the transition so far, the reasons for these problems and suggest possible solutions.

1.2 Purpose

This document is supplied as the Document Master List which provides a directory of all documents which have the status of Draft, Provisional or Approved.

It provides the cross reference to all documents comprising the Transitions in Methodologies project.

1.3 Applicability

This document is an essential reference to all documents supported in the Transitions in Methodologies project.

1.4 Definitions

NDM Pitfalls and guidelines in the transition to Object Oriented software design methodologies.

OO Object Orientation

1.5 Audience

The audience for this document comprise the following:

- project developer
- MSc supervisor
- external reviewers
- members of SEAL
- Head of the Department, Electrical Engineering
- Individuals who will perform internal and external surveillance audits of the SEAL Quality Management System

1.6 Applicable Documents

1.6.1 Standards

a. SEAL QMS Document Layout, Presentation and Typesetting Guide, QS 003, Revision 1.00, 3 October 1994.

1.7 Assumptions

It is assumed that the reader is familiar with the ISO 9000 series standard for quality systems management. It is also assumed that the reader has a basic knowledge of the methodologies used in Software Engineering.

1.8 Requirements Traceability

This document addresses the following requirements:

a) ISO 9001 (1994) 4.5 Document and Data Control

b) ISO 9001 (1994) 4.16 Quality Records

1.9 Procedures

1.9.1 Entering data in the Summary Information File (Alt FI)

a) Title: Enter the name of the document being created

b) Subject: Enter the name of the project or abbreviate the name of the project QMS

c) Author: The name of the person creating this file

d) Keywords: The document code and serial number

e) Comments: The document revision number.

1.9.2 Document Front Page

a) Document project title: Instantiate from Summary Information File data SIF) using left mouse button to select field, click on right mouse button to bring up menu, select 'update field' and click on that item. The field selected will then be automatically updated with the data in the SIF.

b) Document title: Select field and instantiate from SIF.

c) Management/Technical Product: Edit to read Technical or Management Product.

d) Version: Select field and instantiate from SIF.

e) Document Status: Edit to read Draft, provisional or approved.

1.9.3 Using the Configuration Control Table

All elements of the table are instantiated from the SIF.

1.9.4 Using the Document History Table

a) Version: The revision number of the new document

b) Date: The date on which the new revision was created.

c) Status: The status of the created document

d) Who: The author of the updated revision

e) Saved as: the file name (only - no path) of the new document

1.9.5 Using Document Revision Table

a) Version: The revision number of the current document

b) Date: The date on which this revision was updated.

c) Changes: A short description of the nature and location of the changes to the document

1.9.6 Entering details in Master Document List Table

The List is used as follows:

a) Document Name: A descriptive name for the document

b) Document Number: A unique serial number for the artefact

c) Revision Date: The date on which the artefact was modified, or entered into the system, as appropriate.

d) Document Status: For Management and Technical Products this will be Draft, Provisional or Approved. For records this field a '-' is entered since records are not subjected to revision.

e) Date approved: This will be the date of the Management Review meeting.

f) Minute reference: The date of the review meeting and the section of the minutes in which approval for the artefact was recorded.

g) File reference: If the document is in electronic format, the full file path and document name is entered, starting from the SEAL project number as the root. If the document is available in hard copy format only, the term 'Hard copy' is entered.

1.9.7 Revision Control of this document

a) When this document is created from QST 001-10 it is assigned a Revision of 0.01.

b) Once it is approved by the appropriate authority it is raised to Rev 1.00

c) After each internal audit the revision level is raised by a minor point i.e. following the first audit the revision number will be raised to 1.01. (This allows the MDL to be used to record the document baseline to be recorded immediately preceding the audit.)

d) In a one-person project the Project Initiation Audit is used to raise the document to 1.00 status.

e) For each revision change a new file is created.

f) Between document revisions the Change Control element (Revision History) is used to record the changes to the entries in the List, typically in terms of documents (or records) added or updated. These changes will typically refer to new artefacts (records) added in terms of

document number, or the identify of which artefacts have been subjected to updates (technical and management products) This list of changed or updated documents is used to create the entries for the project Document Issue Notices, which are issued periodically to advise clients of the QMS of new or updated artefacts available.

2 Master Document List

Document Name	Document Number	Revision Number	Revision Date	Document Status	Date approved	Minute Reference	File Reference
MANAGEMENT PRODUCTS							
Master Document List	NDM 001	1.01	11/03/98	Approved	98\04\22	2.4	\mp\ndm001.101
Document Creation Template	NDM 002	1.00	11/03/98	Approved	98\04\22	2.4	\mp\ndm002.100
Project Quality Plan	NDM 003	1.00	11/03/98	Approved	98\04\22	2.4	\mp\ndm003.100
Product Description	NDM 004	1.00	11/03/98	Approved	98\04\22	2.4	\mp\ndm004.100
Project Management Plan	NDM 005	1.01	11/03/98	Approved	98\04\22	2.4	\mp\ndm005.101
Configuration Management Plan	NDM 006	1.00	11/03/98	Approved	98\04\22	2.4	\mp\ndm006.100
Contract	NDM 007						At Wits University (hard copy)
Binder Labels	NDM 008	1.00	11/03/98	Approved	98\04\22	2.4	\mp\ndm008.100
Archive Diskette Labels	NDM 009	1.00	11/03/98	Approved	98\04\22	2.4	\mp\ndm009.100
Software Labels	NDM00910	1.00	11/03/98	Approved	98\04\22	2.4	\mp\ndm00910.100
Minutes of meetings Template	NDM 010	1.00	11/03/98	Approved	98\04\22	2.4	\mp\ndm010.100
Document Issue Notice Template	NDM 011	1.00	11/03/98	Approved	98\04\22	2.4	\mp\ndm011.100

Document Name	Document Number	Revision Number	Revision Date	Document Status	Date approved	Minute Reference	File Reference
Call for Review Template	NDM 012	1.00	11/03/98	Approved	98\04\22	2.4	\mp\ndm012.100
Project Issue Report Template	NDM 013						Not applicable
Product Exception Report Template	NDM 014						Not applicable
Inspection and Review Template	NDM 015						Not applicable
TECHNICAL PRODUCTS (Project Overview)							
Project: Front Pages	NDM 100	1.00	11/03/98	Approved	98\04\22	2.4	\tp\reports\ndm100.100
Project Summary/ Overview	NDM 110	1.00	11/03/98	Approved	98\04\22	2.4	\tp\reports\ndm110.100
Project Technical Papers	NDM 120						
Technical Paper: Product Description	NDM 120-10	1.00	11/03/98	Approved	98\10\19	3.	\tp\reports\ndm12010.100
Technical Paper 1	NDM 120-20	1.00	09/30/98	Approved	98\09\23	3.	\tp\reports\ndm12020.100
Technical Paper 2	NDM 120-21	1.00	11/03/98	Approved	98\10\19	3.	\tp\reports\ndm12021.100
Technical Paper: Manuscript (SEAL Style)	NDM 120-30						Not applicable

Document Name	Document Number	Revision Number	Revision Date	Document Status	Date approved	Minute Reference	File Reference
Technical Paper: Manuscript (Journal\Conference style)	NDM 120-31						Not applicable
Technical Paper: presentation slides (White background)	NDM 120-40						Not applicable
Technical Paper: Presentation slides (Colour)	NDM 120-50						Not applicable
Higher Degree: Discussion and Conclusions	NDM 130	1.00	11/03/98	Approved	98\10\19	3.	\tp\reports\ndm130.100
Literature Study	NDM 131	1.00	11/03/98	Approved	98\06\26	3.	\tp\reports\ndm131.100
Higher Degree: Learning from the project	NDM 140	1.00	11/03/98	Approved	98\10\19	3.	\tp\reports\ndm140.100
Project: Bibliography and references	NDM 150	1.00	11/03/98	Approved	98\06\26	3.	\tp\reports\ndm150.100
Higher Degree: Research Assessment	NDM 160	1.00	11/03/98	Approved	98\10\19	3.	\tp\reports\ndm160.100
TECHNICAL PRODUCTS *(Project specific)*							
Summary of Informal Interviews	NDM 300	1.00	11/03/98	Approved	98\05\06	3.	\tp\ndm300.100
List of companies	NDM 301	1.00	11/03/98	Approved	98\05\06	3.	\tp\ndm301.100
Telephonic Questionnaire	NDM 302	1.00	11/03/98	Approved	98\05\06	3.	\tp\ndm302.100

Document Name	Document Number	Revision Number	Revision Date	Document Status	Date approved	Minute Reference	File Reference
Project Proposal	NDM 303	1.00	11/03/98	Approved	98\04\22	2.4	\tp\ndm303.100
Final Questionnaire	NDM 304	1.00	11/03/98	Approved	98\06\08	3.	\tp\ndm304.100
Conclusions from informal interviews	NDM 305	1.00	11/03/98	Approved	98\04\22	2.4	\tp\ndm305.100
Results from Telephonic Interviews	NDM 306	1.00	11/03/98	Approved	98\06\01	3.	\tp\ndm306.100
Conclusions from Telephonic Interviews	NDM 307	1.00	11/03/98	Approved	98\09\23	3.	\tp\ndm307.100
Record of final interviews	NDM 308	1.00	11/03/98	Approved	98\09\23	3.	\tp\ndm308.100
The research process	NDM 309	1.00	11/03/98	Approved	98\09\23	3.	\tp\ndm309.100
Conclusions from final interviews	NDM 310	1.00	11/03/98	Approved	98\09\23	3.	\tp\ndm310.100
Overseas Questionnaire	NDM 311	1.00	11/03/98	Approved	98\09\23	3.	\tp\ndm311.100
Record of overseas interviews	NDM 312	1.00	11/03/98	Approved	98\09\23	3.	\tp\ndm312.100
TECHNICAL PRODUCTS (Project Web support)							

Document Name	Document Number	Revision Number	Revision Date	Document Status	Date approved	Minute Reference	File Reference
Project personnel information page	NDM 900		11/03/98				\www\ndm900.html
Personal CV	NDM 901	1.00	11/03/98	Approved	98\04\22	2.4	\www\ndm901.100
Image(s) of project member(s)	NDM 902		11/03/98				\www\ndm902.jpg
Project Service(s) Page(s)	NDM 920		11/03/98				Not applicable
Project Product(s) Page(s)	NDM 930		11/03/98				Not applicable
Project Information Page(s)	NDM 940		11/03/98				Not applicable
Image 1	www402		11/03/98				\www\www402.gif
Image 2	www420		11/03/98				\www\w 20.gif
Image 3	www445		11/03/98				\www\www445.gif
Image 4	www450		11/03/98				\www\www450.gif
QUALITY RECORDS (Correspondence)							
E-mail distribution list for stakeholders	NDM 1000	1.00	11/03/98	Approved	98\04\22	2.4	\qa\ndm-cgm

Document Name	Document Number	Revision Number	Revision Date	Document Status	Date approved	Minute Reference	File Reference
Correspondence items							\qa\corres\
Mail from ACM	NDM 1001	1.00	11/03/98	Approved	98\04\22	2.4	\qa\corres\980128ac.mvr
Mail from Prof Walker	NDM 1002	1.00	11/03/98	Approved	98\04\22	2.4	\qa\corres\980128aw.mvr
Mail to Prof Walker	NDM 1003	1.00	11/03/98	Approved	98\04\22	2.4	\qa\corres\980128mv.ajw
Mail from supervisor	NDM 1004	1.00	11/03/98	Approved	98\04\22	2.4	\qa\corres\980219bd.mvr
Mail from Prof Walker	NDM 1005	1.00	11/03/98	Approved	98\06\26	3.	\qa\corres\980424aw.mvr
Mail from Prof Walker	NDM 1006	1.00	11/03/98	Approved	98\06\26	3.	\qa\corres\980430aw.mvr
Mail from USA	NDM 1007	1.00	09/19/98	Approved	98\09\23	3.	\qa\corres\980825dl.mvr
Mail from USA	NDM 1008	1.00	09/19/98	Approved	98\09\23	3.	\qa\corres\980821jk.mvr
Mail from USA	NDM 1009	1.00	09/19/98	Approved	98\09\23	3.	\qa\corres\980722wp.mvr
Mail from USA	NDM 1010	1.00	09/19/98	Approved	98\09\23	3.	\qa\corres\980814ea.mvr
QUALITY RECORDS (Agenda's and Minutes)							
Agenda	NDM 2000	1.00	11/03/98	Approved	98\02\02	2.1	\qa\minutes\980202-1.agd
Minutes of meeting	NDM2001	1.00	11/03/98	Approved	98\02\27	2.2	\qa\minutes\980202-1.min
Agenda	NDM2002	1.00	11/03/98	Approved	98\02\27	2.1	\qa\minutes\980227-1.agd
Minutes of meeting	NDM2003	1.00	11/03/98	Approved	98\04\22	2.2	\qa\minutes\980227-1.min
Agenda	NDM2004	1.00	11/03/98	Approved	98\04\22	2.1	\qa\minutes\980422-1.agd
Minutes of meeting	NDM2005	1.00	11/03/98	Approved	98\05\06	2.2	\qa\minutes\980422-1.min

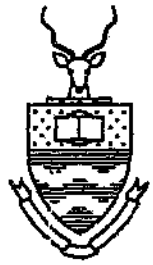
Document Name	Document Number	Revision Number	Revision Date	Document Status	Date approved	Minute Reference	File Reference
Agenda	NDM2006	1.00	11/03/98	Approved	98\05\06	2.1	\qa\minutes\980506-1.agd
Minutes of meeting	NDM2007	1.00	11/03/98	Approved	98\05\11	2.2	\qa\minutes\980506-1.min
Agenda	NDM2008	1.00	11/03/98	Approved	98\05\11	2.1	\qa\minutes\980511-1.agd
Minutes of meeting	NDM2009	1.00	11/03/98	Approved	98\06\01	2.2.	\qa\minutes\980511-1.min
Agenda	NDM2010	1.00	11/03/98	Approved	98\06\01	2.1	\qa\minutes\980601-1.agd
Minutes of meeting	NDM2011	1.00	11/03/98	Approved	98\06\08	2.2	\qa\minutes\980601-1.min
Agenda	NDM2012	1.00	11/03/98	Approved	98\06\08	2.1	\qa\minutes\980608-1.agd
Minutes of meeting	NDM2013	1.00	11/03/98	Approved	98\06\26	2.2	\qa\minutes\980608-1.min
Agenda	NDM2014	1.00	11/03/98	Approved	98\06\26	2.1	\qa\minutes\980626-1.agd
Minutes of meeting	NDM2015	1.00	11/03/98	Approved	98\08\21	3.	\qa\minutes\980626-1.min
Agenda	NDM2016	1.00	11/03/98	Approved	98\06\26	3.	\qa\minutes\980821-1.agd
Minutes of meeting	NDM2017	1.00	11/03/98	Approved	98\09\23	3.	\qa\minutes\980821-1.min
Agenda	NDM2018	1.00	11/03/98	Approved	98\09\23	3.	\qa\minutes\980923-1.agd
Minutes of meeting	NDM2019	1.00	11/03/98	Approved	98\10\19	3.	\qa\minutes\980923-1.min
Agenda	NDM2020	1.00	11/03/98	Approved	98\10\19	3.	\qa\minutes\981019-1.agd
Minutes of meeting	NDM2021	1.00	11/03/98	Approved	98\10\19	3.	\qa\minutes\981019-1.min
QUALITY RECORDS (Audit reports)							

Document Name	Document Number	Revision Number	Revision Date	Document Status	Date approved	Minute Reference	File Reference
Audit report 1	NDM 3000	1.00	11/03/98	Approved	98\08\26	3.	\qa\auditrep\9814pi.par
Audit report 2	NDM 3001	1.00	11/03/98	Approved	98\09\23	3.	\qa\auditrep\9814ip01.par
Audit report 3	NDM 3002	1.00	11/03/98	Approved	98\10\19	3.	\qa\auditrep\9814ip02.par
QUALITY RECORDS (Document Issue Notices)							
Notice of files placed on the file server for the project	NDM 4000	1.00	11/03/98	Approved	98\04\22	2.4	\qa\notices\980303-1.din
Notice of files placed on the file server for the project	NDM4001	1.00	11/03/98	Approved	98\06\26	3.	\qa\notices\980626-1.din
Notice of files placed on the file server for the project	NDM4002	1.00	11/03/98	Approved	98\09\23	3.	\qa\notices\980923-1.din
QUALITY RECORDS (Call for Review)							
Call for reviews of documents for this project	NDM 5000	1.00	11/03/98	Approved	98\04\22	2.4	\qa\reviews\980306-1.cfr
Call for reviews of technical papers	NDM 5001	1.00	11/03/98	Approved	98\09\23	2.4	\qa\reviews\980821-1.cfr
Call for review of documents	NDM 5002	1.00	11/03/98	Approved	98\04\22	2.4	\qa\reviews\980306-2.cfr
Call for review of documents	NDM 5003	1.00	11/03/98	Approved	98\04\22	2.4	\qa\reviews\980306-3.cfr
Call for review of documents	NDM 5004	1.00	11/03/98	Approved	98\04\22	2.4	\qa\reviews\980306-4.cfr

Document Name	Document Number	Revision Number	Revision Date	Document Status	Date approved	Minute Refer-ence	File Reference
Call for review of documents	NDM 5005	1.00	11/03/98	Approved	98\04\22	2.4	\qa\reviews\980306-5.cfr
Call for review of documents	NDM 5006	1.00	11/03/98	Approved	98\04\22	2.4	\qa\reviews\980306-6.cfr
Call for review of documents	NDM 5007	1.00	11/03/98	Approved	98\04\22	2.4	\qa\reviews\980306-7.cfr
Call for review of documents	NDM 5008	1.00	11/03/98	Approved	98\06\26	3.	\qa\reviews\980506-1.cfr
Call for review of documents	NDM 5009	1.00	11/03/98	Approved	98\06\26	3.	\qa\reviews\980506-2.cfr
Call for review of documents	NDM [illegible]	1.00	11/03/98	Approved	98\06\26	3.	\qa\reviews\980608-1.cfr
Call for review of documents	NDM 5011	1.00	11/03/98	Approved	98\06\26	3.	\qa\reviews\980622-1.cfr
Call for review of documents	NDM 5012	1.00	11/03/98	Approved	98\06\26	3.	\qa\reviews\980622-2.cfr
Call for review of documents	NDM 5013	1.00	11/03/98	Approved	98\09\23	3.	\qa\reviews\980921-1.cfr
Call for review of documents	NDM 5014	1.00	11/03/98	Approved	98\09\23	3.	\qa\reviews\980921-2.cfr
Call for review of documents	NDM 5015	1.00	11/03/98	Approved	98\10\19	3.	\qa\reviews\981016-1.cfr
Call for review of documents	NDM 5016	1.00	11/03/98	Approved	98\10\19	3.	\qa\reviews\981016-2.cfr
QUALITY RECORDS (Project Issue reports)							
Records of problems associated with the project Quality management system	NDM 6000						Not applicable

Document Name	Document Number	Revision Number	Revision Date	Document Status	Date approved	Minute Reference	File Reference
QUALITY RECORDS (Product Problem Reports)							
Records of problems associated with the product developed on the project	NDM 7000						Not applicable
QUALITY RECORDS (Inspection and Review Records)							
Records of the application of software inspection and reviews process applied to a document	NDM 8000						Not applicable
QUALITY RECORDS (Backup and archives)							
Backup Register	NDM 9000						Part of Configuration Management Plan NDM006
QUALITY RECORDS (Financial Records)							
A record of financial transactions on the project	NDM 10000						Hard copy
QUALITY RECORDS (Software Product Test Records)							

Document Name	Document Number	Revision Number	Revision Date	Document Status	Date approved	Minute Reference	File Reference
A record of software tests conducted using the Product test specification and checklist template	NDM 11000						Not applicable



NDM QMS

Project Management Plan

Management Product

Version 1.01

Document Status: Approved

Table of Contents

Table of Contents ii

Change History iv

Configuration Control iv

Document History iv

Revision History iv

Management Authorisation iv

Change Forecast iv

1 Scope 1

1.1 Introduction 1

1.2 Purpose 1

1.3 Definitions 1

1.4 Audience 1

1.5 Applicable Documents 2

1.6 Assumptions 2

1.7 ISO 9001 Requirements Traceability 2

2 Product Quality Plan 4

3 Project Management Plan 5

3.1 Overview 5

3.2 Product Development Team 5

3.3 Roles and Responsibilities 5

4 Project Control 7

4.1 Planning 7

4.2 Reviewing 7

4.3 Reporting 8

4.4 Document control 8

4.5 Change Control 8

5 Product Development Plan 9

5.1 Dependencies 9

5.2 Required Human Resources 9

5.3 Available Human Resources (may be supported as NDM 005-30) 9

5.4 Human Resource allocation (may be separately supported as NDM 005-40) 9

5.5 Hardware and Software Resources. (may be separately supported as NDM 005-50) 10

5.6 Training Needs Identification. (may be separately supported as NDM 005-60) 11

6 Budget (May be supported as NDM 005-20) 12

7 Work Breakdown Structure, Obligations and Schedule (May be supported as XX 005-10) 16

Change History.

Configuration Control

Project:	NDM QMS
Title:	Project Management Plan
Doc. Reference:	C:\MSC\MP\NDM005.100
Created by:	M Jansen Van Rensburg
Creation Date:	12 February 1998

Document History

Version	Date	Status	Who	Saved as:
0.01	98\02\12	Draft	MVR	NDM005.010
1.00	98\04\22	Approved	MVR	NDM005.100

Revision History

Version	Date	Changes
0.01	98\02\12	New document created using QST12410.103
1.00	98\04\22	Approved
1.00	98\06\22	Updated the WBS
1.01	98\06\22	Prof Walker corrected the footers.
1.01	98\09\23	Added additional tasks in WBS

Management Authorisation

Version	Date	Status	Management Board Minute Reference
1.00	98\04\22	Approved	98\04\22 2.4

Change Forecast

This document will be updated each time there is a change to one of the tables in this Plan i.e. resources, task element, or schedule.

1 Scope

1.1 Introduction

This Plan provides an overview of the required resources to produce the Transitions in Methodologies research report. It will form the terms of reference for all contributing parties during the project.

- This Plan outlines the human resources requirements.
- It provides a list of tasks to be performed and a time scale.
- It provides a means for capturing important project metrics regarding estimated and actual effort to perform allocated tasks.

The costs associated with the development of the product are outside the scope of this Plan.

1.2 Purpose

The purpose of this Plan is to:

- Present a clear statement of all deliverables.
- Present a clear statement of work allocation and responsibilities to be undertaken by all parties.

Present details of resources associated with this product.

1.3 Definitions

NDM Pitfalls and guidelines in the transition to Object Oriented software design methodologies

1.4 Audience

The audience for this document comprise the various stakeholders of the SEAL, including:

- full-time staff members of the SEAL
- All full-time and part-time post-graduate students associated with the SEAL

- Members of the SEAL Management Board
- Head of the Department, Electrical Engineering
- Individuals who will perform internal and external surveillance audits of the SEAL Quality Management System.
- project developer

1.5 Applicable Documents

1.5.1 Specifications

1.5.2 Standards

a. SEAL QMS Document Creation Template, QS 002, Revision 0.01, 5 April 1994.

b. SEAL QMS Document Layout, Presentation and Typesetting Guide, QS 003, Revision 0.01, 1 April 1994.

1.5.3 Procedures

1.5.4 Guidelines

1.5.5 Other Documents

1.6 Assumptions

None

1.7 ISO 9001 Requirements Traceability

a. ISO 9001 (1994) 4.1 Management Responsibility

b. ISO 9001 (1994) 4.2 Quality System

c. ISO 9001 (1994) 4.3 Contract Review

d. ISO 9001 (1994) 4.4 Design Control

e. ISO 9001 (1994) 4.5 Document and Data Control

f. ISO 9001 (1994) 4.8 Product Identification and Traceability

g. ISO 9001 (1994) 4.9 Process Control

h. ISO 9001 (1994) 4.10 Inspection and Testing

i. ISO 9001 (1994) 4.13 Control of Non-conforming Product

j. ISO 9001 (1994) 4.14 Corrective and Preventative Action

k. ISO 9001 (1994) 4.16 Quality Records

l. ISO 9001 (1994) 4.18 Training

2 Product Quality Plan

The team associated with this project is totally committed to quality and all system documentation produced as part of the product will be reviewed internally by the Review team and produced in accordance with the standards and formats associated with the SEAL QMS.

The Quality Plan associated with this project is identified as NDM 003, as defined in QS 195 SEAL QMS Project Documentation and Support Standard.

3 Project Management Plan

3.1 Overview

The aim of this Project Management Plan is to define the structure of this project and to create the technical products.

This resource plan must be formally approved by the Product Manager before work on the Technical products is undertaken.

3.2 Product Development Team

The development team comprises:

a. The Product Manager of the Transitions in methodologies project.

b. The developer(s) of this product / service.

3.3 Roles and Responsibilities

3.3.1 Product Developer

a. Establish and maintain a product plan for developing this product / service.

b. Ensure all Technical Exceptions are properly reported.

c. Prepare Detailed Work Plans as necessary.

d. Prepare and present regular Checkpoint Reports to the Project Manager.

e. Take direction from the Project Manager for matters related to the project.

f. Act as the Project Editor for the product.

3.3.2 Product Manager

a. Monitor progress and resource utilisation of the Product Team, and initiate corrective action where necessary.

b. Identify required resources and make these available to the team.

c. Act as the focal reporting point for the product

d. Ensure that Product Reviews are held as planned..

e. Interface to the SEAL Management board.

f. Be responsible for the timely delivery of the product.

g. Give direction to the Product development team.

4 Project Control

4.1 Planning

The planning of different tasks of this project will be performed by the Product developer under the approval of the Product Manager.

Any detailed plan will not be executed without the approval of the Product Manager.

4.2 Reviewing

a. All draft versions of the product will be reviewed by the Product Manager.

b. In addition, certain versions which are so specified in the Work Breakdown Structure will be reviewed by the members of other product groups.

4.2.1 Planned Arrangements for Reviews

All project artefacts (i.e. Management, Technical, and Quality Assurance Products) are subject to review. Management and Technical Products may be subject to a number of revisions before a product is approved by the Product Manager. QA Products are not normally changed after being produced and are simply approved by the Project Manager. (Changes are only made if there are factual errors in the first revision.)

The obligations of Project Manager and Product Develop(s) associated with each review activity are listed in the Work Breakdown Structure, Obligations and Schedule (Section 7).

4.2.2 Scheduling of Reviews

Each Management and Technical Product identified in the Work Breakdown Structure (Section 7) will be reviewed by the Product Manager. This will normally take place in a review meeting. These meetings will be listed in the Project Schedule, and typically identified with the product(s) (or WBS item(s)) under review.

4.2.3 Records of Reviews

Records of review meetings are maintained as minutes taken at the meetings. These review meeting minutes will be listed as entries in the QA Products in the Project Master Document List (NDM 001).

4.3 Reporting

Control is affected by both monitoring and reporting upon progress to management and members of the team.

4.3.1 Responsibilities of the Product Developer

The product developer(s) will report to the Product Manager at intervals defined in the Minutes of Product Review Meetings detailing:

a. Progress to date

b. Effort required to complete

c. Problems encountered and action taken

4.3.2 Responsibilities of the Product Manager

a. The Product Manager will initiate corrective action as necessary to keep the project on schedule.

b. The Product Manager will report the progress of the project to the SEAL Management Board.

4.4 Document control

Document control will be performed as per SEAL QMS policy and procedures:

a. QS125 SEAL QMS Document and Data Control

b. QS 195 SEAL QMS Project Documentation Standard

c. In accordance with the requirements of QS 195, this document will be numbered as NDM 005.

d. The Management and Technical products identified in the WBS in Section 7 will be reflected in the Master Document List for this project (NDM 001).

4.5 Change Control

Any work outside of the Product Description (NDM 004) will be formally approved by the Project Manager.

5 Product Development Plan

5.1 Dependencies

5.1.1 Internal dependencies

Project development progress will depend on the developer's time available.

5.1.2 External dependencies

Since a request for funding has been submitted to the FRD, the items that will require external funding will be delayed until funding has been granted.

5.2 Required Human Resources

The following resources are required to develop the product:

Number	Resource	Commitment
1	Product Manager	100% throughout the project
1	Product Developer(s)	100% throughout the project

5.3 Available Human Resources (may be supported as NDM 005-30)

The following human resources are available as on 1 February 1998:

Who	Contact Details(postal address, phone, fax, e-mail addresses)	Role
Miranda Jansen van Rensburg	011 471 2929	student / project developer
Prof Barry Dwolatzky	011 716 5368	supervisor / project supervisor

5.4 Human Resource allocation (may be separately supported as NDM 005-40)

Who	Project Role	Responsible for tasks
Prof Dwolatzky	Project Leader	• Maintain project management plan • Maintain work breakdown structure • Develop agenda for meetings • Build requirements verification and validation register

Who	Project Role	Responsible for tasks
		• Chair team meetings.
M van Rensburg	CM Co-ordinator	• Version management of all documents on file server. • Configuration status accounting • Configuration management plan • Creation and issue of document issue notices • Maintain master document list • Checking for viruses in incoming documents.
M van Rensburg	QA Co-ordinator	• Capture and manage minutes of meetings • Document control, copying of documents prior to project meetings • Maintain project binders • Maintain product description • Assemble audit reports and supervise corrective actions • Manage disposition of quality system and product problem reports • Manage Call for Reviews
M van Rensburg	Technical Document Editors	• Build and maintain technical documents emerging from the software development lifecycle
Prof Barry Dwolatzky	Course Supervisor\ Client	• Maintain Quality Plan • Approval of management and technical products

5.5 Hardware and Software Resources. (may be separately supported as NDM 005-50)

The following hardware and software resources are required:

Hardware/software items	Quantity	Date required	Date supplied
PC	1	98\02\01	98\02\01
FTP / Internet connection	1	98\02\01	98\03\01
Printer	1	98\02\01	98\03\01
Word processor	1	98\02\01	98\02\01

5.6 Training Needs Identification. (may be separately supported as NDM 005-60)

The following training is required for this project:

Project Member	Course Title	Offered by	Course date	Attended?
M Jansen van Rensburg	SEAL 98 -03 Principles of ISO 9001	SEAL	98\02\09 - 98\02\10	Yes
M Jansen van Rensburg	SEAL 98 - 04 Managing software project using ISO 9001	SEAL	98\02\11 - 98\02\13	Yes

6 Budget (May be supported as NDM 005-20)

WBS Task	Description	Labour Cost	Equip. Cost	Material Cost	Office Space	Support Staff	Training Cost	Total
1.1	Master Document List	0	0	R100	0	0	R1000	R1100
1.2	Document Creation Template	0	0	0	0	0	0	0
1.3	Quality Plan	0	0	0	0	0	0	0
1.4	Product Description	0	0	0	0	0	0	0
1.5	Project Management Plan	0	0	0	0	0	0	0
1.6	Configuration Management Plan	0	0	0	0	0	0	0
1.7	Binder Label	0	0	0	0	0	0	0
1.8	Archive Diskette Labels	0	0	0	0	0	0	0
1.9	Minutes of meeting template	0	0	0	0	0	0	0
1.10	Document Issue Notice template	0	0	0	0	0	0	0
1.11	Call for review template	0	0	0	0	0	0	0
2.1	Higher Degree Project: Project Front Page	0	0	0	0	0	0	0
2.2	Project Summary	0	0	0	0	0	0	0
2.3	Project Technical Papers	0	0	0	0	0	0	0

2.4	Higher Degree: Discussion and Conclusions	0	0	0	0	0	0	0
2.5	Literature Study	0	0	R400	0	0	0	R400
2.6	Higher Degree: Learning from the project	0	0	0	0	0	0	0
2.7	Project: Bibliography and references	0	0	0	0	0	0	0
2.8	Higher Degree: Research Assessment	0	0	0	0	0	0	0
2.9	Summary of informal interviews	0	0	0	0	0	0	0
2.10	List of companies	0	0	0	0	0	0	0
2.11	Telephonic Questionnaire	0	0	R500	0	0	0	R500
2.12	Project Proposal	0	0	0	0	0	0	0
2.13	Final Questionnaire	0	0	0	0	0	0	0
2.14	Conclusions from Informal interviews	0	0	0	0	0	0	0
2.15	Results from Telephonic Interviews	0	0	0	0	0	0	0
2.16	Conclusions from Telephonic Interviews	0	0	0	0	0	0	0
2.17	Record of final	0	0	0	0	0	0	0

	Interviews							
2.18	The research process	0	0	0	0	0	0	0
2.19	Final interviews: conclusions	0	0	R3000	0	0	0	R3000
2.20	Overseas questionnaire	0	0	0	0	0	0	0
2.21	Record of overseas interviews	0	0	R15000	0	0	0	R15000
3.1	E-mail distribution list for stakeholders	0	0	0	0	0	0	0
3.2	Correspondence items	0	0	0	0	0	0	0
3.3	Minutes and agendas of meetings	0	0	0	0	0	0	0
3.4	Audit reports	0	0	0	0	0	0	0
3.5	Notices of files placed on the file server for that project	0	0	0	0	0	0	0
3.6	Call for reviews of documents for this project	0	0	0	0	0	0	0
3.7	Records of reviews of technical papers	0	0	0	0	0	0	0
3.8	Records of problems associated with that project Quality management system	0	0	0	0	0	0	0

3.9	Records of problems associated with the product developed on that project	0	0	0	0	0	0	0
3.10	Records of the application of software inspection and reviews process applied to a document	0	0	0	0	0	0	0
3.11	Backup Register	0	0	0	0	0	0	0
3.12	A record of financial transactions on the project	0	0	0	0	0	0	0
Totals		0	0	R19000	0	0	R1000	R20000

7 Work Breakdown Structure, Obligations and Schedule (May be supported as XX 005-10)

Project Artefact (WBS Item)	Doc. No	Who	Obligations		Effort (hrs)		Start Date		End Date	
			Developer	Manager	Est.	Act	Sched-uled	Actual	Sched-uled	Actual
<artefact name i.e. Management or Technical Product>	<project doc. no.>	<who>	<what tasks must the developer perform?>	<what tasks must the Manager perform?>	<no.>	<no.>	<yy\mm\dd>	<yy\mm\dd>	<yy\mm\dd>	<yy\mm\dd>
1. Management Products										
1.1. Master Document List	NDM 001	MVR	Create	Review	20	20	98\02\12	98\02\12	98\02\28	98\03\01
1.2. Document Creation Template	NDM 002	MVR	Create	Review	10	8	98\02\12	98\02\12	98\02\13	98\02\28
1.3 Quality Plan	NDM 003	MVR	Create	Review	20	30	98\02\12	98\02\12	98\02\28	98\03\02
1.4 Product Description	NDM 004	MVR	Create	Review	20	20	98\02\12	98\02\12	98\02\28	98\03\03
1.5 Project Management Plan	NDM 005	MVR	Create	Review	20	20	98\02\12	98\02\12	98\02\22	98\0304
1.6 Configuration Management Plan	NDM 006	MVR	Create	Review	20	20	98\02\12	98\02\12	98\02\28	98\02\28
1.7 Binder Label	NDM 008	MVR	Create	Review	2	2	98\02\12	98\02\12	98\02\12	98\02\12
1.8 Archive Diskette Labels	NDM 009	MVR	Create	Review	2	2	98\02\12	98\02\12	98\02\12	98\02\12

Project Artefact (WBS item)	Doc. No	Who	Obligations		Effort (hrs)		Start Date		End Date	
			Developer	Manager	Est.	Act	Sched-uled	Actual	Sched-uled	Actual
1.9 Minutes of meeting template	NDM 010	MVR	Create	Review	2	2	98\02\12	98\02\12	98\02\12	98\02\12
1.10 Document Issue Notice	NDM 011	MVR	Create	Review	2	2	98\02\12	98\02\12	98\02\12	98\02\12
1.11 Call for review template	NDM 012	MVR	Create	Review	2	2	98\02\12	98\02\12	98\02\12	98\02\12
2. Technical Products										
2.1 Higher Degree Project: Project Front Page	NDM 100	MVR	Create	Review	8	8	98\02\13	98\02\13	98\02\16	98\02\16
2.2 Project Summary	NDM 110	MVR	Create	Review	8	8	98\02\12	98\02\12	98\02\12	98\02\12
Project Technical Papers	NDM 120									
2.4 Technical Paper: Product Description	NDM 120-10	MVR	Create	Review	40	20	98\07\01	98\07\01	98\09\01	98\09\01
2.5 Technical Paper: South African situation	NDM 120-20	MVR	Create	Review	40	80	98\07\01	98\07\01	98\09\01	98\09\01
2.6 Technical Paper: International situation	NDM 120-21	MVR	Create	Review	40	60	98\09\01	98\09\16	98\09\21	98\09\21

Project Artefact (WBS Item)	Doc. No	Who	Obligations		Effort (hrs)		Start Date		End Date	
			Developer	Manager	Est.	Act	Sched-uled	Actual	Sched-uled	Actual
2.7 Technical Paper: Manuscript (SEAL Style)	NDM 120-30	MVR	Create	Review	40		98\06\01		98\09\01	
2.8 Technical Paper: Manuscript (Journal\Conference style)	NDM 120-31	MVR	Create	Review	40		98\06\01		98\09\01	
2.9 Higher Degree: Discussion and Conclusions	NDM 130	MVR	Create	Review	20	20	98\06\01	98\09\02	98\09\01	98\09\16
2.10 Literature study	NDM 131	MVR	Create	Review	200	200	98\02\01	98\02\01	98\09\09	98\09\16
2.11 Higher Degree: Learning from the project	NDM 140	MVR	Create	Review	10	10	98\06\01	98\09\20	98\09\21	98\09\21
2.12 Project: Bibliography and references	NDM 150	MVR	Create	Review	40	40	98\05\01	98\06\16	58\09\01	98\08\25
2.13 Higher Degree: Research Assessment	NDM 160	MVR	Create	Review	10	10	98\10\01	98\10\01	98\10\18	98\10\18
2.14 Summary of Informal Interviews	NDM 300	MVR	Create	Review	20	20	98\03\30	98\04\20	93\04\20	98\04\22
2.15 List of companies	NDM 301	MVR	Create	Review	20	30	98\04\30	98\04\30	98\05\05	98\05\05
2.16 Telephonic Questionnaire	NDM 302	MVR	Create	Review	40	40	98\04\30	98\04\30	98\05\05	98\05\05

Project Artefact (WBS item)	Doc. No	Who	Obligations		Effort (hrs)		Start Date		End Date	
			Developer	Manager	Est.	Act	Sched-uled	Actual	Sched-uled	Actual
2.17 Project Proposal	NDM 303	MVR	Create	Review	10	20	97\10\03	97\10\03	97\10\05	97\10\08
2.18 Final Questionnaire	NDM 304	MVR	Create	Review	10	20	98\05\30	98\05\30	98\05\30	98\05\30
2.19 Conclusions from informal interviews	NDM 305	MVR	Create	Review	10	10	98\04\10	98\04\10	98\04\10	98\04\10
2.20 Results from Telephonic Interviews	NDM 306	MVR	Create	Review	20	20	98\05\31	98\05\31	98\05\31	98\05\31
2.21 Conclusions from Telephonic interviews	NDM 307	MVR	Create	Review	20	20	98\08\01	98\08\01	98\08\20	98\08\20
2.22 Record of Final Questionnaire	NDM 308	MVR	Create	Review	40	40	98\08\01	98\08\02	98\09\21	98\09\21
2.23 The research process	NDM 309	MVR	Create	Review	10	20	98\03\01	98\03\01	98\03\03	98\03\03
2.24 Final interviews: conclusions	NDM 310	MVR	Create	Review	20	20	98\08\01	98\08\01	98\08\20	98\08\20
2.25 Overseas questionnaire	NDM 311	MVR	Create	Review	20	20	98\08\01	98\08\01	98\08\20	98\08\20
2.26 Record of overseas interviews	NDM 312	MVR	Create	Review	40	20	98\09\16	98\09\16	98\09\20	98\09\20

Project Artefact (WBS Item)	Doc. No	Who	Obligations		Effort (hrs)		Start Date		End Date	
			Developer	Manager	Est.	Act	Sched-uled	Actual	Sched-uled	Actual
2.27 CV for web page	NDM 901	MVR	Create	Review	8	8	98\06\01	98\04\01	98\06\02	98\04\02

A Literature Study

Pitfalls and guidelines in the transition to object oriented software design methodologies

M Jansen van Rensburg

Software Engineering Applications Laboratory, Electrical Engineering,
University of the Witwatersrand, Johannesburg, South Africa

Summary:

This document contains a literature study that investigates the literature currently available on the transition to OO (Object Orientation). It provides an overview of the issues involved in the adoption of OO. It consists of a summary of technology itself, the benefits associated and then goes on to describe the situation by looking at the human resource ues, then the corporate issues and lastly the technical issues. Finally it also investigates what the future holds for OO.

1. Introduction

Due to the nature of the software engineering industry there is a constant move to new strategies for solving design problems. More specifically there is a move towards the Object Oriented methodologies, presumably because of the various advantages offered in terms of maintainability of code produced this way. As with various other aspects of the software industry there are however also problems encountered in this transition and lessons to be learned from the experience of companies who have already performed this change.

There is therefore a need to know what these problems are and how to avoid them in order to make a success of OO projects, so that the advantages offered by OO can indeed be utilised.

2. Discussion

The document summarises the research that has already been done in object orientation, so that the lessons learnt by others can provide some insight for the future. It also provides guidelines for companies who are currently contemplating a change to the OO methodologies, covering important issues one should know about prior to this change.

It will also summarise the problems faced in the transition so far, the reasons for these problems and suggest possible solutions. Lastly it will investigate any new trends in the OO arena.

3. Introduction

3.1 Background

In 1995, Fayad [28] commented that the move to OO has become a serious managerial and technical decision rather than the academical exercise it used to be. Recently, (1997) Love [53] also added that "ten years ago, objects were old news to a few researchers, a few entrepreneurs and even fewer commercial organisations ... however, it was the SIGS publications and the ACM OOPSLA conference that began to seriously spread the word in 1986 and 1987."

Furthermore, according to the Gartner Group[1], "by 2000, at least half of all new applications will use object technology for user interfaces and complex client server functionality."

We can therefore see that the move to object orientation has been and still is substantial.

[1] Taken from the SoftwareFutures web page, http://www.softwarefutures.com

3.1.1 Where did it start

According to Graham[33], the concept of objects has entered the computer science arena in 3 distinct ways:

- From the programming language community, requiring new development techniques for simulation, user interface development etc.
- From database theorists in need of abstract models. The models developed dealt with abstract objects and inheritance but focused on the data aspects of entities, ignoring the procedures that entities could perform
- The artificial intelligence community went one step further by introducing frames and attaching procedures to attributes.

Viljoen[90] found that the need for object orientation developed due to current operating environments that are

- Based on components
- Overlapping in functionality
- Consisting of incompatible components and using hybrid components
- Poorly integrated
- Using ill define protocols
- Containing redundancy
- Fragmented

He summarises the history of object orientation as given in Table 3.1

Table 3.1

1967	Introduction of Simula-67 by Kristen Nygaar
1969	Development of Smalltalk by Alan Kay
1980	OO concepts introduced into languages such as Pascal and Ada.
1985	Bjarne Stroustrup introduces C++ as a formal hybrid language. Steve Jobs develops the NeXT operating system under C++.
1987	Smalltalk released for DOS, OS/2 Windows and Apple Mac Zortech releases C++ for DOS, UNIX, and OS/2
1992	Borland is repositioned as a prime OO technology (C++, Pascal etc.)

Graham[33] found it difficult to believe that two mere technical benefits namely encapsulation and inheritance are sufficient to explain the sudden success of OT (Object Technology). According to him, it took 25 years before anyone in mainstream IT(Information Technology) took any notice.

Although improvements in the power of hardware could provide some explanation, he reasoned that it was the political economy of the situation that dictated the swing. Businesses needed to be more adaptable in a changing world, and it also became necessary to improve the productivity of both developers and users.

3.1.2 Why Objects?

Before going further one needs to summarise the objectives of this technology that has seemingly taken the world by storm:

Moreau[62] characterises OO systems as supporting:

- Data and procedure encapsulation: both data and procedures operating on the data are

private, objects interact only through well defined interfaces

- Typed message passing: the procedure invoked in response to a message can be made dependent on the type of the argument of the message
- Late binding of messages to target modules where association between messages and their receivers are resolved at runtime
- The inheritance mechanism

Viljoen[90] further defines the objectives of object orientation as providing:

- a formalised model of reality
- a more natural paradigm for emulating the real world
- understandability
- reusability
- open endedness of software components for maximum configurability
- natural concurrency, since OO seems to produce better solutions for problems involving real time processing
- workload balancing by allowing objects to serve multiple masters
- reduced life cycle cost by increasing programmer productivity and reducing maintenance costs
- system robustness
- maintainability
- a rationalised organisation
- modularity for robustness
- interchangeability
- architectural elegance
- system quality

3.1.3 Achieving the OO goal

To determine how object oriented a system is, one can refer to Viljoen's[90] degrees of "object orientedness" described in Table 3.2:

Table 3.2

Degree 1	Usage of common ideas
Degree 2	Usage of common subroutines
Degree 3	Usage of common data structures
Degree 4	Reusability of program code
Degree 5	Modularity of code and data
Degree 6	Abstraction mechanisms in languages
Degree 7	Code and data encapsulation
Degree 8	Inter-application portability of objects
Degree 9	Inter-platform portability of objects
Degree 10	Reusability through classification
Degree 11	Dynamic binding of objects
Degree 12	Late binding of objects

Alternatively, in Table 3.3 Page-Jones[72] defines seven levels of OO expertise:

Table 3.3

1	Innocent (has never heard of OO)
2	Aware (has read an article about OO)
3	Apprentice (has attended a 5 day course)
4	Practitioner (ready to use OO on a real project)
5	Journeyman (uses OO naturally in his job)
6	Master (has internalised the details of OO; knows when he can break the rules)
7	Expert (writes books, gives lecturers etc.)

3.2 The "Fad of the year"

Before discussing object orientation further, one needs to consider this phenomenon.

3.2.1 According to Meller Page-Jones

Page-Jones[71] reports that this occurrence has seen many shapes:

In 1984 The Fad was "relational stuff"; in 1986, artificial intelligence; in 1988, CASE tools; and in 1989 repositories. He concludes that the 1990s have been plagued by object orientation, being the "Fad of the Decade".

Figure 3.1 shows the *natural* adoption curve for a technology.

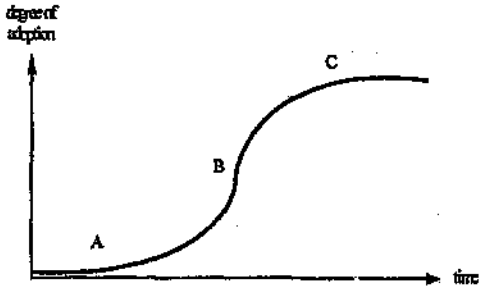


Figure 3.1 The natural adoption curve for a typical technology

It starts as experimental technology (label "A"), then reaches commercial maturity and, if feasible, is adopted rapidly ("B"). Eventually, a limit of useful applications is reached and its adoption levels off ("C"). (The horizontal time-scale is of the order of one or two decades even in this fast-moving business.)

Figure 3.2 shows the adoption curve for a technology that becomes a "Fad of the Year".

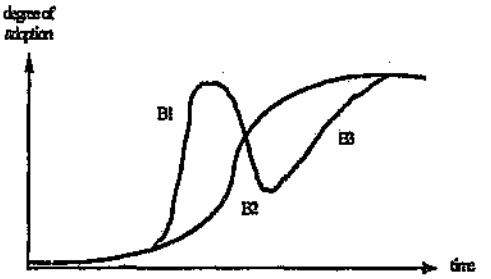


Figure 3.2 The distorted adoption curve for a "Fad of the Year"

The first distortion (marked "B1") results from the direct "Fad of the Year" bandwagon effect as everyone tries to adopt the technology. The second distortion (marked "B2") occurs as people get disillusioned with the new technology. Unfortunately this occurrence also often puts off potential serious adopters.

"B3" denotes how, in time, the technology recovers as the more mature and circumspect users persevere with its use.

The author argues that since object orientation has been a Superfad through the 1990s, we can review what has happened since 1990 and also make some predictions about what might happen during the rest of the decade. During 1997 and 1998 the author found that publications and conferences devoted to object orientation had increased dramatically.

3.2.2 According to Graham

Graham[33] uses a similar graph (Figure 3.3) to illustrate how a new technology is accepted.

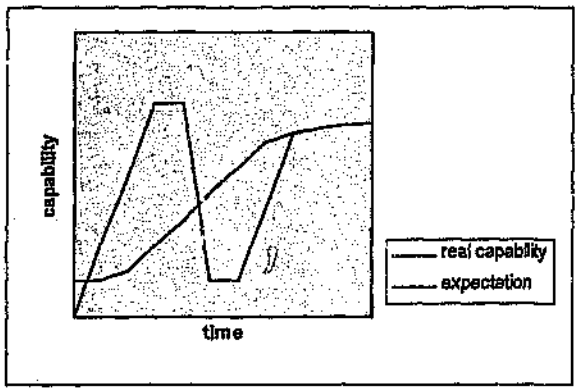


Figure 3.3 Capacity versus time

Early users' first exposure often leads to disappointment. Expectation decreases while the real capability in the mean time increases so that the two curves cross over. This phenomenon implies that vendors will charge more for services and products and that buyers will pay more to gain a competitive advantage from the technology and to get trained.

3.3 The situation then and now

Graham's analysis reflects that market trends are upward and changing in character[33]:

3.3.1 1990

Users were attending awareness briefings and gathering information[33]

3.3.2 1991

Graham[33] reports that a few mission critical projects using OO had begun.

One can also refer to Taylor's[43] comment in 1991 that "nearly 50% of Fortune 500 companies are developing OO applications intended for deployment .."

Philip [43] reported a lack of infrastructure as a reason why OO was not accepted faster. He also described the situation by saying that companies were happy with the (then) present situation, since they were getting benefits and were not interested in disclosing how they handled their competitive advantage.

3.3.3 1992

Graham[32] reported that the market for training courses in object oriented programming and methods began to mature and advanced users were completing their first pilot projects.

An article on tunable formalism[16] appeared in this year, promising to help move OO software development closer to becoming an engineering

discipline. This method makes it possible for theoreticians and practitioners to use the same model, since users can work with different levels of formalism ranging from informal to mathematically rigorous.

3.3.4 1993

Litvintchouk [51] commented that the Software Engineering Institute (SEI) estimated the period from conceptualisation of a technology to popularisation to be 18 years. It was felt that OO will be no different in this matter than the N other technologies already investigated.

According to Graham [32] OO was beginning to be used in large-scale mission critical projects in data processing.

Advanced users had completely switched to object oriented development for new systems while continuing to maintain conventional legacy systems.

The then current buyers of object technology were large software houses, smaller software houses and end users. The large software houses mainly used OT to gain productivity from reuse. The smaller software houses used OT to develop complex products such as CASE tools, gaining productivity from reuse and using the technology to enter new markets. End users were building pilots, buying mentoring services and showing an interest in methodologies, their main applications being client server systems and GUIs. Very few had switched over to OO totally.

3.3.5 1994

According to Pickering's survey[74], the situation in 1994 could be described as given in Table3.4.

Yourdon [92] also gives the 1994-1996 expectations (Figure 3.4) for OO.

Table 3.4

Technology	Projects used on (%)		Success (%)		Effective penetration (%)	
	1991	1993	1991	1993	1991	1993
OO	2.8	11.9	91.7	66.3	3.5	7.9
Structured methods	71.4	60.7	90.2	84.3	64.4	51.2
CASE	28.8	26.3	59.7	70.8	17.2	18.6

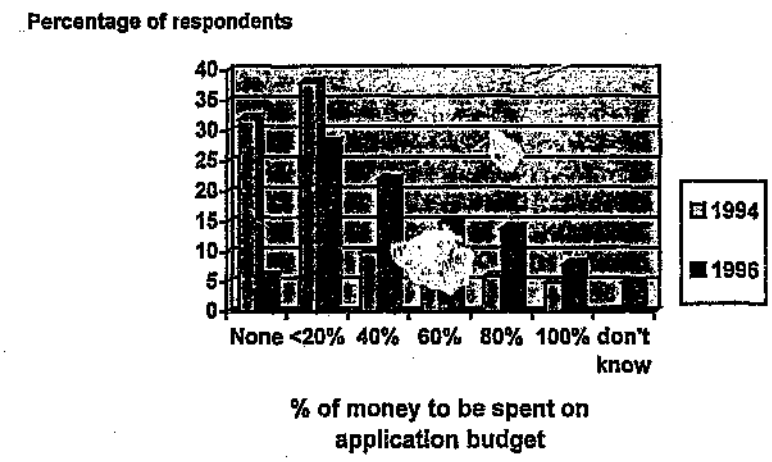


Figure 3.4

At this time[32] (1994) Graham believed that the technology was near the peak of his expectation curve. He felt that crossover would occur within two years, since the then latest market surveys (from Ovum for example) forecast very rapid growth in the OO market but predicted a flattening out in the growth rate within two years. Also, since it takes about two years for an organisation to understand and adopt a new approach, then (1994) would be the right time for users to begin education and training programmes to be prepared for the crossover.

3.3.6 1997 / 1998

The 1997 Cutter Consortium's report[34], for which more than 200 enterprises world-wide were used in a survey, concluded that:

- companies are adopting OO to achieve increased productivity through reuse
- to achieve this increase in productivity, it was found that 3-5 years is necessary to change the way in which software is developed as well as to acquire an OO infrastructure.
- Most organisations use consultants and invest heavily in training
- Early transitions to OO in large companies have resulted in several failures. However, most IT managers feel confident that they can develop OO applications successfully
- Most companies have started developing some form of component libraries, and 40% of these companies already have frameworks
- Microsoft is the largest source of OO classes
- UML is the most popular OO notation
- C++ and Java are the preferred OO languages. Microsoft Visual Studio is the most popular C++ development environment.
- 70% of the companies surveyed have developed at least one large OO application.
- 41% of companies reported at least 1 failure, reasons given being poor management and a shortage of experienced developers.
- there is a lag in the adoption of OO databases - companies still use relational databases.

3.3.6.1 Training

The Cutter Consortium Report[34] concluded that 39% of organisations rely on developers to decide which projects will be OO based. The report also found that companies acquire the OO developers they need as follows:

51% train current staff in OO, 41% recruit staff with OO skills, 8% use consultants.

The January OMO survey[67] found that 42.1% of respondents had some form of classroom training in their primary methodology, whereas 57.1% had not.

On the OMO survey's[67] question of the number of object technology related conferences participants have attended, the results obtained were: (Table 3.5)

Table 3.5

None	39.5% of respondents
1	19.7%
2-5	32.9%
5-10	5.3%
More than 10	2.6%

3.3.6.2 Methodology

During the January OMO survey [67], respondents gave the following answers regarding their primary object methodology: (Table 3.6)

Table 3.6

UML	43.8% of respondents
OML	4.1%
OMT	15.1%
Booch	8.2%
Coad/Yourdon	1.4%
Shlaer/Mellor	0%
OOSE/Jacobson	8.2%
None	16.4%
Other	2.7%

When asked which phases were formally defined and use a methodology, the respondents in the January 1998 OMO survey [67] replied with the results as given in Table 3.7.

Table 3.7

Analysis	24.0%
Architecture	23.5%
High Level Design	24.0%
Detailed design	19.0%
Testing	9.5%

According to the Cutter Consortium Report[34], UML (by now officially approved by the OMG) is already used by 15% of companies. The results were as follows: OMT - tation (34%), Booch (15%), Use cases (15%). Most companies indicated a move to UML during 1998.

3.3.6.3 *Languages*

The January 1998 survey in Object Magazine[67], found the primary programming language among respondents to be as given in Table 3.8:

Table 3.8

C++	49.3%
Java	18.7%
Smalltalk	16%
Eiffel	2.7%
Ada9X	1.3%
Visual Basic	4%
Powerbuilder	1.3%
Object Database	0%
Relational	1.3%
Other	5.3%

3.3.6.4 *Budget*

According to the survey in the Object Magazine in January 1998 [68], the total annual budget for object-oriented products and services is as shown in Table 3.9.

Table 3.9

< $10000	18%
$10000-99999	35%
$100000-499999	18%
$500000-999999	2%
$1 million or more	27%

These results came from the industries given in Table 3.10.

Table 3.10

Consulting	15%
Services (computer related)	14%
Finance/banking/accounting /insurance	12%
Government	8%
Healthcare	8%
Other	8%
R&D/education	8%
Telecommunication	8%
Manufacturing (computer related)	7%
Utilities/transportation	6%
Manufacturing (non computer related)	3%
Legal	1%
Wholesale/resale	1%
Aerospace	0%
Services(non computer related)	0%

3.3.6.5 *Internet*

According to the Cutter Consortium[34] the Internet has had the effect of pushing large numbers of companies into the exploration of object technology. (78% of the companies agreed that the Internet has led to an increase in OT interest).

3.3.6.6 Data storage

The Cutter Consortium [34] reported that 50% of companies are not using OO or object relational database to store data.

For large companies, 8% of companies planned to acquire an OO or Object Relational database product, and 18% did not. Regarding small companies, 24% of companies planned to acquire an OO or Object Relational database product, and 33% did not.

3.4 Reasons for moving to OO

3.4.1 The wrong reasons

Page-Jones[71] pointed out that "unfortunately, object orientation, though a valuable and sophisticated approach to constructing software, has been over-marketed by unscrupulous vendors. Worse, in some shops the term "object orientation" has become little more than a mantra, chanted by gullible managers who mindlessly seek instant salvation from the software woes around them."

According to Page-Jones [72] reusability, robustness, etc. are the politically correct reasons to move to OO. In support of this he discusses the ten wrong reasons why managers adopt OO.

- Having no direction, companies fall for the Fad of the year phenomenon, and will embrace every new technology
- Companies are told about the tremendous benefits but not of the dangers
- Attempts to please top management
- Competitors are using OO, thereby posing a threat
- Programmers want C++ on their CVs, because they do not want to be left behind. Companies often feel pressured to comply to those needs.
- The inspiration of speakers is not a valid reason, since the problems in the company's unique situation should be evaluated carefully to determine whether OO's advantages will solve them.
- Structured techniques were inadequate – where structured methods were successful, OO will most likely also be, since the company has vision, discipline and management commitment.
- C++ is a better C – as prescribed by the revolutionary approach this is not the best way to learn OO concepts.
- Use of a purchased OO library could be expensive.
- New development tools forces a move to OO, however, OO remains a difficult software technique and integration could be expensive.

3.4.2 The right reasons

3.4.2.1 Higher-quality systems

Object orientation yields higher-quality systems than traditional approaches do, which leads to robust systems, reliability, extensibility, maintainability and usability. [71]

"Furthermore, the repeated use of classes from libraries will render each class very sturdy, as any initial problems become worked out early in each class's lifetime."

3.4.2.2 Higher quality development process

"Structured analysis contains an historical schism between modelling process and modelling data that was never quite healed." [71] Although there is an overwhelming *proliferation* of object-oriented development methods, for most companies, even a middle-of-the-road method represents a quality improvement over their erstwhile software practices. This unexpected advantage, namely improving the *quality* of the development process, also leads to an improvement in the *productivity* of development.

According to Bulman[22], a well-developed OO process of software development improves on a structured process in every development phase:

- Maintenance is improved due to the encapsulation of objects.
- With proper OO analysis, both design and implementation becomes simpler due to the more complete separation modules.
- Better partitioning provides an opportunity for groups of classes to be developed in parallel.
- Implementation becomes easier via inheritance and due to the fact that the objects themselves will be more reusable.

At Digital Consulting [49] the following resulted:

- Introducing OO formalises documentation methods since a complete problem definition allows for the correct setting of customer and engineering expectations,
- Schedules and cost estimates are forecasted more accurately.

3.4.2.3 Capacity to build larger systems

"For large problems where a large staff is needed OO leads on each facet of the required teamwork Once the initial definition of classes is started, teams can start development in separate groups of classes with less ripple effect between classes when changes are needed. The granularity of development tasks leads to a natural strong fire wall between OT and most other classes." [22]

3.4.2.4 Solution to a business problem

To promote OO with management it is necessary to present OO as a solution to a business problem, according to Litvintchouk.[51]

In contrast to this, Hildenberg [43] states that " clients want cost effective high quality solutions to meet their needs, and they want them now. OOT must be thought of as a tool, not a solution. They don't care what technology is used"

3.4.2.5 Importance for industrial applications:

Pancake commented [73] on OT importance for industrial applications:

- OO makes it possible to model systems that are close in structure to their real world analogues.
- Applying OO in business engineering and software design leads to flexibility and agility.
- the role-oriented nature of OT makes the information hierarchy less compartmentalised.

3.4.2.6 Previous methods are lacking

Ed Yourdon's arguments for OO are the following[92]:

- Problems have been experienced with classical methodologies
- Modern systems are different than when Structured Analysis and Design was developed in the 1970s
- The technology of languages and environments has changed dramatically over the past 20 years

In a recent Usenet discussion[58], OO was regarded as an improvement over structured programming just as structured programming was an improvement over flowcharting. This is because OO is more powerful in decoupling modules than any previous paradigm.

Marty Cagan from IDE (Interactive Development Environments) noted that in structured development there has always been a gap between various levels of specification and implementation. Object-oriented development holds the promise of reducing this gap by providing a smooth transformation between specification and implementation":[76]

However, Ed Yourdon[24] warns that even if OO analysis is more applicable than SA(structured analysis) from a technical perspective, many organisations have too much inertia to switch. He found that new paradigms are generally accepted only when the old paradigm fails to solve new problems with which the organisation is faced. He therefore feels that OO analysis will be more politically acceptable in environments where SA has failed on large, visible projects and also on projects where reusability and graphical user interfaces are seen to be key issues from the onset.

3.4.2.7 Short term vs. long term advantages

According to Hazeltine[43] there are both short-term benefits (productivity increases) and long term benefits that will position the company towards taking advantage of corporate computing.

3.4.2.8 The future

In the Cutter Consortium's report, the ability to take advantage of new operating systems and tools and participation in the future of computing was mentioned. [34]

3.4.2.9 Rapid development

The semantic richness of object models makes specifications more reversible and supports rapid application development directly. Sound object oriented analysis helps to deliver the benefits of OT much earlier in the life cycle. [32] [62]

3.4.2.10 Encapsulation

The advantages of OO in graphical information systems were found to be that OO systems provide strong encapsulation in that objects use private methods to manipulate private data exclusively in response to messages from other objects. [62]

This also leads to improved interfaces.[34]

3.4.2.11 Extensibility and flexibility

Several sources confirmed the role that OO has to play in providing extensible and flexible software:

- "We chose 00 technology primarily because we needed to build a new system that would be flexible in the face · regulation changes and business changes. We also had to be able to adapt to new technologies, especially in the user interface." – Litvintchouk[51]
- Moreau in his article mentions the late binding of messages to target objects which provides flexibility to reconfigure systems dynamically without recompilation. [62]
- "Object-oriented concepts such as reuse and encapsulation offer many benefits to application development, particularly in managing complexity and change" – Shan, in his article about objects and servers[83].

3.4.2.12 Reuse

"The sophistication a company shows in its use of classes and components is a good sign of the company's overall progress toward a OO based development capability" – Harmon [34]

There is currently little empirical information on what to expect from reuse in terms of productivity and quality gains. [9]

Management expectations are often raised with the promise of reusability. However, the first OO project in an organisation will most likely require a large investment without achieving much productivity. A company should therefore be sure of the reasons for choosing OO and then invest sufficiently to realise achieving it - Page-Jones[71]

Page-Jones's experience is that with OO one can "reduce the number of lines of new code for an application by about 5:1. However, this reduction in new code requires the development of a sound in-house library and about 5 years of work."[71]

According to Graham[33] the good news for organisations migrating to OO is that "reuse can

slash development cost and some organisations have doubled productivity annually because of this".

In the study done by Basili et al [9] the following assumptions were investigated:

- high reuse leads to lower likelihood of defects
- high reuse leads to lower rework effort
- high reuse leads to high productivity

The study showed the strong impact on productivity in terms of linear relationships between productivity and reuse rate, which can be used by managers as a baseline for comparison in future studies.

The graphs they produced show a direct equivalent relationship between productivity and reuse (slope 1.11) and an indirect equivalent relationship between rework and reuse ratio.

3.4.3 A panacea?

In an article about moving from C to C++, Ted Goldstein [76] said that "object oriented programming is not a panacea". A discussion is then given about how structured programming was given as a solution in the 80s and how OO should be adopted in the right way in order not to create another "winter"...:

"There is considerable risk involved in making any technology transition, and object oriented technology is no exception. A similar situation occurred during the 1980's to AI programming. Many promises made, but few were kept. In the early '80's, artificial intelligence was supposed to solve the "software crisis". Reality set in and many problems turned out to be harder than expected. People became disillusioned and even those AI techniques, which had validity, were dismissed. This had led to the current state of an "AI winter." There are a number of issues involving languages systems, and libraries which could easily bring on "an object-oriented winter." It is important for managers and practitioners of object-oriented programming to recognise exactly what the technology can accomplish. Myths concerning productivity improvement, language choice, dynamic versus statically typed languages, performance results, and testing must be countered with the reality of experience in object-oriented programming. The reality behind object-oriented programming is more about making a commitment to well designed software, and making the capital investment in training and tools to accomplish the goal of software quality and overcome the software crisis."

Various means for preventing this object winter can be found in the article by Anderson et al,[3] for which groups had to write down the most important steps that companies could take to foster increased architectural competency. The results were:

- a **management** commitment to an architectural approach rather than churning out code
- architecture recognised as a viable and supported **career path**
- time for **learning** through mentoring and interaction
- safe and open **peer relationships**
- the development of a **corporate memory** – making knowledge explicit

The various human issues that were mentioned here will now be discussed in the following sections.

4. The human resource issues

4.1 Management of OO projects

Vaishnavi [43] pointed out management's role by saying that the OO paradigm has to move out into the management and user communities to achieve success.

4.1.1 Commitment from the top

There are numerous references to the importance of management's role in the transition to OO:

- Vayda[89] mentions the importance of commitment from the top and
- Bernsen[23] also says that "management sponsorship and commitment is key"

- Often organisations "undertake OO technology ...without much management support. Because there isn't an atmosphere fostered to make it happen people get resistive attitudes. That causes failure." [51]

- Because the OO technology transition takes a long time and is not an easy transition and because it should be seen as a different way of doing business, management must be supportive and committed, all the way up to the top of the organisation. This is not usually the case: OO technology tends to come in through the back door, from the bottom up. We will see that the NASA Software Engineering Laboratory took 7 years to complete the transition to OO. Other organisations take less time because managers were committed to it. Litvintchouk's research concluded that what really determines the speed of insertion of OO is the approach to technology transition, and to change management, more than the technical aspects. [51]

4.1.2 Different from normal management

Five people give their advice[39]. According to Kenny Rubin from ParcPlace,

"a project that is lead by an "object technology expert," without basic project management skills, is a prescription for failure. Although basic managerial goals remain unchanged when we use object technology, the specific ways in which we achieve the goals are changed. Object technology affects the software development processes, resources, and products. As such, a project lead by an experienced project manager, who does not have an understanding of object technology, is likely to fail due to ignorance surrounding required technical and organisational change. The best managers of object projects have good fundamental project management skills, and appreciate the influence of object technology on software development processes, resources, and products."

According to Coplien from AT&T the differences for OO projects relate to "doing risk management in a field of emerging tools and methods that lack track records available for other methods."

According to Johnson from Rothwell International OO technology changes the cost equations since coding becomes cheaper and more time is spent on design issues. A successful project manager must have knowledge of both project management and cost issues, while experience in OO development helps.

According to Fayad[28], it takes a significant learning curve to bring teams up to a level of competence on OO, which implies a longer initial time to market. It may also require new tools, new programming languages, new metrics and new software development processes. All these issues can complicate the software manager's job. The manager needs to deal with new or different problems: staffing, training, scheduling, cost estimation, standards, documentation, etc.

Finally, Page-Jones prescribes a different sequence of activities (for OO) from the traditional project life cycle. This significantly affects development sequence of a project as well as the cost benefit analysis of the project undertaking.[71]

4.1.3 Guidance

There seems to be little guidance for project managers in the form of published sources. It seems that training should be provided for management as well as developers. [28][29]

Daniels[39] gives the following guidelines for managers:

- Project managers must understand the technology being used on their projects.

- Evolutionary delivery is the best way of reducing project risks.

- Each cycle in the development should be linked to time scales and should deliver some executable code.

- It is never too soon to start with designing the software architecture.

4.1.3.1 The customer

As part of changing the culture the customers need to be convinced, since they often associate a new

technology with high risk.[29] Some steps that are recommended:

- provide capability briefing charts that address the perceived risk areas, such as benefits versus cost, training schedules, CASE tool strategies, and Return On Investment.
- train the customer to be familiar with OO topics
- explain long-term payoffs.

4.1.3.2 Preplanning

"Careful pre-project planning will smooth the transition from a chaotic software development process. The software development plan must be the first document produced." [28]

4.1.3.3 Taylor's suggestions

Taylor [43] suggests a new model for the management of corporate computing:

- It is not sufficient to just modify the waterfall model by giving an OO twist to each phase of the development cycle, and then claiming to have OO analysis, OO design and OO programming in place.
- Software should be built in layers similar to the way hardware is built. Therefore frameworks should be used.
- He also feels that OO cannot be adopted by just adopting the new technology. One has to change the organisation, corporate culture and reward structure.

4.1.3.4 Knowledge engineering

To help support AT&T in their move to OO they made use of knowledge engineering[43]. Basic research was necessary to manage object technology. Empirical research was necessary to collect the data that managers need for cost models and resource planning. Behavioural and organisational research was necessary to determine how to make reuse happen.

4.2 Teamwork

Beck[31] wrote that human communication is still the most important remaining barrier to programmer productivity. According to Booch, "attempting to develop a complex object-oriented software system with a non-object oriented team will add significant risk to the success of any project" since the roles in an OO project are very different from a non-OO project. He also reasons that there are some roles in an OO project which do not even have analogies in non-OO projects.

According to Coplien, communication is the key to effective teams: "with the advent of objects came the promise of well-documented interfaces that communicated behavioural intent while hiding implementation detail. Experience has shown that design is more subtle than that." Therefore it seems that many of these issues require good communication and therefore good teamwork.

Recommendations for a successful development team[29] include working with process improvement groups to get the team involved with documenting the new processes and initiating participative management.

4.2.1 Staffing

The following advice is documented:

- Hazeltine[43] recommends high-level architect support for the development team - a ratio of at least 1 architect to 5 developers.
- According to Vayda,[89] choosing the right team implies choosing the right attitude rather than technical skills, which can be learned.
- Team buy-in: both Vayda[89] and Fayad [28] found that developers must therefore be selected carefully, to possess aggressiveness to push forward new ideas and receptiveness to gain knowledge.
- De Champeaux[23] recommends using a central object support team.

- Page-Jones[71] warns against staffing projects entirely with stage threes (as described in section 3.1.3 Achieving the OO goal)

- It is recommended to get an OO expert that can solve methodology limitations not uncovered during training, train new employees, adapt CASE tools and do extensions for the problem domain and the organisation's standards.[29]

- Having a domain analyst that can co-ordinate with domain experts on different projects, develop specifications and designs for reuse and create or maintain a class library is recommended.[29]

- It is also advisable to have a prototyping expert that can evaluate requirements for completeness and design efficiency, prototype objects and communicate requirements to software developers. [29]

- "The most important factor in getting OO technology inserted was skill leverage....the good procedural designers became the good object designers, probably because they had abstract reasoning capability"[51]

- Organisations often have to choose between traditional developers with many years of experience and newcomers with skills in OT and modern methods. Neither will do since they should learn from each other- Graham. [33]

4.3 Training of OO professionals

4.3.1 Why training is important

"What we learn from OO technology and its changes we need to apply to the other areas of business because it is clear that the business world will have to cope with increasing amount of change. Training is key" – Hazeltine.[43]

Page-Jones wrote about a popular myth about object orientation where companies claimed "We don't need any requirements-analysis training; we're object-oriented."[71] He argued that successful object orientation required more training in general software-engineering principles, because the structures of object orientation are more complex than those of a structured design systems. He reasoned that a structured system contains lines of code and modules which implies two levels of construction whereas an object-oriented system contains lines of code, methods and classes resulting in three levels of construction. Inheritance hierarchies also adds further complexity. He mentioned his concern about the frenzy among training companies seeing an opportunity, which lead to object orientation becoming undisciplined.

That training is a critical issue, is confirmed by Pancake[73]: Most dollars spent in converting to OT can be contributed to education and training. (There is the paradigm shift, as well as new programming language, software development techniques and tools to get familiar with)

Stroustrup[66] commented that better education should be a top priority, and should consist of a blend of theory and practice.

The role of training was also emphasised by Janet Conway of GE Advanced Concepts Centre,[56] who claimed that the largest cost in converting to object oriented technology is training.

According to Taylor[43], a study covering a 50-year span illustrated a return on investment for training being 42 times that of capital equipment. In spite of this large return, the total dollar amount of investment for training by American companies has been only a small fraction of that for equipment. He concluded that there needs to be better support for and more investment in education and training to be able to survive.

One can summarise this section about the importance of training, by using Lato's [47] comment: *"Technologies come and go. In time the brand new technology of today will be a dinosaur. But a company that has learned how to learn will be able to adapt to the ever changing reality of doing business today."*

4.3.2 Guidelines

Several authors gave their guidelines for training:

New developers should learn from the experience of others. However, this does not imply handholding or outsourcing. [89]

Fayad [29] prescribes five W's: What - qualified OO training, Where – in-house, Who – everyone involved in the project from customer to tester, When - just in time, Why - rapid change in culture.

To foster an environment where new technologies thrive, Latol[47] provides the following guidelines for organisations:

- Cultivating early adopters in-house who are enthusiastic about the technology and willing to stay ahead.
- Using mentoring is a good way of transferring knowledge from experienced people
- Giving time for experimenting with the new technology before making it part of developers' normal workload.
- The technology should be introduced at a comfortable pace.
- People need to be constantly reminded of the benefits of using OO.
- Management should be trained on the technology to be effective at managing people and the project. They need to understand the deliverables, the resources needed, differences compared to the old way of doing things, and risks associated with transitioning to the new technology. Without proper training management will either fear new technology or can get the wrong perception of the technology.
- Investing in tools, consulting and conferences, enables companies to benefit from industry experience. Using consultants can provide an external check on the way work is done which can reduce the associated risk.
- Since many people will accept outside expertise more easily than in-house, buying outside experience often benefits the process.

Dodani's article [27] takes a so-called shock therapy approach as a solution to the problem of immediately retraining highly qualified people in the workplace.

Still on retraining procedure-oriented developers: research done by Manns and Nelson[56] found that analogies can help procedure oriented developers in the transition to OO. Their case study was done using people with procedure oriented experience. Some of their findings can be summarised as follows:

Table 4.1

OO concept	Associated procedure oriented concept
Message passing	Function parameter passing, spaghetti code
Class hierarchy	Database diagrams, flowcharts, structure charts
Object abstraction and encapsulation	Abstraction of procedures
Object	Program, module, function, database, database record, data structure, library, attribute, file, abstract data type
Class	Database, database table, data structure
Class hierarchy (inheritance)	Nested data structures, functional decomposition, hierarchical data model, database normalisation
Methods	Function , procedure
Encapsulation and information hiding	Linking object code, local variables
Polymorphism	Case statement / if then else statements

Bulman[22] recommends the order of training to be managers, analysts, designers, and then programmers. He also mentions that the probability that many organisations will follow this path is very low.

Yourdon [92] refers to Page-Jones's experience levels[72] when giving his training suggestions

- He recommends assessing current expertise level and establishing medium and long term goals.
- A crucial project should not be attempted with only stage 3 developers and below.
- In-house expertise in fundamental methodologies and principles should be nurtured.

Manns [16] recommends the following:

- A project immediately after training to maintain interest and skills obtained.
- A structured mentoring program to provide support.
- A system for locating OO literature.
- A plan for in-house versus contact courses with the prospect of educating some employees to become future trainers.

According to D'Souza[18] the following steps should be taken:

- The transition should be done on multiple fronts, including consulting, mentoring, and formal training.
- A language training course is not enough.
- The curriculum should follow consistent object principles throughout.
- The course should actively encourage reuse by providing and requiring component reuse.

Pancake adds the following:[73]

- Problems arise when people attempt to learn OT through a hybrid language. The author argues that C++ is the worst language to start with as it allows you to slip back into the habits you're used to, whereas languages like Smalltalk and Self force you to think in an object oriented way.
- Pancake reasons that it is probably better to defer language training until the OO concepts are understood, since the learning hurdle is good design.
- It is difficult to evaluate the quality of a good OO design, which again demonstrates the need for training.
- There is a lack of good didactic examples based on real world needs instead of the usual vending machine or ATM examples.
- Garbage collection plays an important role for people using a hybrid approach in changing. Training should therefore cater for this.

Gabriel raised the following technical issues:[18]

- The student needs to unlearn defining data structures separately from the control mechanisms that manipulate them.
- One should guard against problems that occur in an uncontrolled OO design where messages fan out so that communication becomes complex.
- Training should include teaching about limiting complexity by keeping co-operative components within a framework.

4.3.3 Selecting a training organisation

The following criteria should be used when selecting an external training service [11]:

- The training company's client base.
- The skill level of the instructors.
- The quality of the course material.
- Whether an integrated curriculum is provided, i.e. whether architecture is taken into account, whether only one OO language is taught, etc.
- Whether the company takes advantage of new technology tools, e.g. the World Wide Web.
- How easily the company's technology can be leveraged.

- Whether the training program can be customised to fit the company's specific needs.
- Whether the training program makes use of more than one method such as lectures, hands on exercises, one-on-one tutorials, etc.
- Whether support and mentoring is provided throughout the project.
- The level of follow up that is provided.

4.3.4 A new curriculum

It is Meyer's [60] opinion that OO should be taught as early as possible since it provides good preparation with formal approaches to software specification and verification. However, he warns against the introduction of a formal specification method as it could overwhelm students.

In his article Wick [91] provides a description of the content of a first year computer science course and also describes how it should be adapted so that students learn that OO is purely a medium and not the message. The message should be software reusability.

"The major pitfalls of current approaches to using C++ and OO in CS1 stem from a misplaced focus on the tools rather than on the application of the tools...Students are not motivated through the use of concepts before they are asked to consider their implementation". He also reasons that life cycle issues such as maintenance and documentation should be made important, arguing that CS1 courses often only teach students to be consumers not producers of reusable software.

4.3.5 Profile of an OO student

Liu [52] describes the profile of an OO student:

Several hundred students in introductory OO programming courses were studied in 1990. They used Smalltalk to learn OO and a smaller study was also done on students using C++. Students varied from being experienced to novices at programming, varied from managers to software development people, etc. He found the following:

- Inquisitive students and those with broad programming backgrounds did better.
- A strong and perhaps unexpected association arose between C experience and object learning. The authors explained this saying that people with C experience are likely to be those with the software sophistication to be successful with objects.
- More positive results when learning objects came from students with more recent programming experience.
- Negative findings came from students with experience in assembler or COBOL.

4.4 Organisational issues

From the opinions and experiences of the following people it is clear that the organisational issues have a large part to play in the successful implementation of OO in a company.

In Graham's opinion [32][33] the problems remaining are organisational. It has to come from the top.

The focus is on changes in the organisation as it adopts OO, with regards to the project team structure, management policies, development process, etc. – Korson [43].

Korson (director of COMSOFT - Consortium for the management of Software Technology) also said that he has seen "projects fail to achieve the promise of OOT because the organisation did not understand the changes in corporate infrastructure that were necessary to support the object paradigm".

Johnson [39] reasoned that good OO systems are structured differently than conventional systems, and therefore a different organisational structure is needed. He recommended teams owning groups of related classes and individuals within teams owning classes.

Hazeltine[43] from NCR agreed that the organisation design is extremely important for

success, and is as much a factor as the object oriented paradigm itself. He recommended tailoring the organisational design to a specific organisation, project and number of people.

The implication of staff reassignment can be described as follows: If an OO-experienced company has to write only 20% of the new lines of code per application that it used to, then the company needs only 20% of its former programmers. There are several new positions that need to be filled. Therefore, the "extra programmers" should be re-trained and re-assigned into roles such as library manager, library class programmers, prototyper, requirements analyst, implementation designer, etc. - Page-Jones.[71] And finally, the role of the organisation was confirmed in Mentor Graphics[43], a company that adopted OO on a corporate wide basis in 1985. In this case class development was centralised but the components of the framework were created and managed at department level. This was made possible because of decentralisation within organisations.

Table 5.1

Author	Inheritance	Multiple inheritance	Attributes for objects	Aggregation	State Transition Diagrams	Events	Event Traces	Integrated tool set
Ballin				✔		✔	✔	
Coad	✔	✔	✔	✔	✔	✔		
Colbert	✔	✔	✔	✔	✔			✔
Edwards						✔	✔	✔
Gibson						✔		
Jacobson	✔	✔	✔		✔	✔	✔	✔
Kurtz	✔	✔		✔	✔	✔	✔	
Odell	✔	✔	✔	✔	✔	✔	✔	
Page-Jones	✔	✔	✔		✔	✔	✔	
Rumbaugh	✔	✔	✔	✔	✔	✔	✔	
Shlaer-Mellor	✔	✔	✔		✔	✔	✔	
Wirfs-Brock	✔	✔	✔	✔				

5. Corporate Issues

5.1 Time when Undertaking OO

Would an organisation attempting OO today make faster progress?

Fayad [51] argued that in the past, there were very few tools and methods available. However, a lot more became available to choose from which often lead to confusion about what to use.

Table 5.1, describing the 1992 situation[21], supports Fayad's reasoning. Tick marks indicate features supported by the specific methodology.

At the time, the great diversity of methodologies was being advocated as a healthy sign as the technology was still developing.

Page-Jones[71] recently said the following: "Although just about everything we need for object orientation is here, nothing is as sophisticated as it will be five to ten years from now. Some of the

facilities that we now have are awkward to use and there are many reports of development problems... On the other hand, most of these problems are receding into the past, as object orientation, recently a revolutionary new approach to software, is now joining the mainstream of software techniques. A shop currently embarking upon object orientation is therefore no longer an early adopter and is no longer forced to navigate a pioneering voyage to Terra Incognita."

One can therefore derive that the situation is now more favourable for companies currently contemplating the transition to OO.

5.1.1 Standards

Regarding object standards Seldon[82] described the situation as follows: "OO languages began to appear in the mid 70s. By 1994 there were over 50 languages. As can be expected users could not find one language that catered for all their needs and this lead to the methods war, which saw various methodologies begin to incorporate each others, techniques. Unfortunately the methods that eventually emerged had strengths and weaknesses. The subsequent development of the Unified Modelling Language (UML) began in Oct 1994."

The UML has since been accepted by the Object Management Group (OMG) late in 1997.

Many argued that OO has taken a long time to mature in the enterprise due to the lack of standards. Hopefully this new unified approach will provide the solution. With one standard it should now be a lot easier to move to OO than it was in 1992.

5.1.2 Maturity

Page-Jones[71] refers to Watts Humphrey's description of how an organisation's sophistication goes through five levels, or ages, of software-engineering maturity in applying OO. According to Humphrey's description there are the five ages: Anarchy, Folklore, Methods, Metrics and Engineering.

The Age of Anarchy implies "a maelstrom of methods, with all the analysts and programmers doing their jobs in whatever manner they please." The Age of Folklore is about informal knowledge about what works. The Age of Methods formalises approaches to building systems and increases the likelihood of success by leaving less to chance. The Age of Metrics introduces quantitative measurements of processes. The Age of Engineering entails producing systems becoming routine.

The author prescribes that a shop needs to be at least in the Age of Methods before it can maturely absorb, manage and exploit object orientation. This correlates with the need for quality in the development process.

Ed Yourdon mentions the following critical areas for measuring the maturity or readiness for using Object Technology in an organisation.[92]

- Is a support infrastructure available?
- Are good OO tools and implementation technologies available?
- Is the organisation sophisticated enough to successfully change its development methods?
- Are applications being developed by the organisation the kind that will effectively use the OO paradigm?

Concerns about the maturity of the technology, arising from industry can be derived from a survey,[74] summarising reasons given by 1991 and 1993 survey respondents for NOT using OO at the time. (Table 5.2) Results indicate percentages of respondents.

Table 5.2

REASON	1991	1993
Not aware of technology	31.0	13.2
Benefits not demonstrated	3.5	19.3
No business need	17.2	3.5
Technology too costly	0.9	2.6
Organisation unprepared	19.8	19.3
Technology too immature	19.8	36.8
Other	7.8	5.3

5.2 First Project

In Lato's article[47] the importance of pilot projects is emphasised and the following guidelines provided.

5.2.1 Expecting mistakes

"Mistakes are a normal part of the learning process so ensure the project can afford to make mistakes and take the time to learn it right. The pilot project should be done in a short interval so it provides timely feedback. Understanding and applying the technology correctly is more important than meeting the schedule" – Lato[47]

Page-Jones [71] recommends accepting that the first project will not yield great financial dividends, and might wind up *costing* money. The project team will take time to learn, and new tools for development and library management will require a monetary investment. He cautions against building up unrealistic expectations and against trying to adopt every last aspect of object orientation all at once.

5.2.2 High visibility

Korncoff[23] recommends choosing a small non-critical problem to establish an initial success. According to Vayda[89] the project should have high visibility but simultaneously also a lenient schedule.

5.2.3 Budget

The budget for introducing object orientation should not be tied to a single project budget.[71]

5.2.4 A new project

According to Fayad[28] the first project must be a new project which does not have added issues such as legacy systems.

5.2.5 Large and meaningful

The first project must be large and meaningful enough to influence the other project members' attitudes towards OO.[28]

5.2.6 Support

The first project must be supported by the customer and high level management.[28]

5.2.7 Staffing

Staffing the first OO project requires special consideration. People selected for this team should be eager to learn new concepts.[28]

5.2.8 Management

When promoting the concept to management, one should not refer to a paradigm shift as this holds a negative connotation. OO can be promoted by developing metrics, standards for languages and libraries of success stories.[23]

5.3 Adoption of a corporate object technology centre

Korson's concern is about the need for adaptation to change since OO is still changing.[43]

"C++ has gone from single inheritance to multiple inheritance, exception handling and template types...What will happen to a culture within a corporation with this accelerated pace of change? Many companies are used to hiring COBOL programmers ands then using them for 10-20 years with little need to update their knowledge. How do you see the infrastructure of an organisation adapting to support continued change?"

Korson[44] also found that corporations adopting object-oriented technology on a large scale are facing some complex technical, cultural, organisational, management, and policy issues.

To facilitate this adoption, a number of organisations have set up a corporate "object technology centre" (OTC).

According to Vaishnavi, [88] having an OTC can help as a change agent, in establishing policies, in bringing members of the technical staff into personal interaction with mentors, etc.

5.3.1 Goals of an OTC

The following goals have been established:

- Technology insertion through hot lines, documentation etc.
- Promoting general interest in the technology from the bottom up.
- Getting top down support for investment.
- Providing recommendations on standards, tools etc.
- Providing training.
- Ensuring the methodology promotes true software engineering.
- Driving a common understanding amongst all technology practitioners.
- Managing rapid development.

5.3.2 Roles of an OTC

- Education / knowledge broker: book library, newsletters, educational services, speakers.
- Technology transfer: mentoring, training, pilot projects, apprenticeships.
- Technology support: hot lines, customising the technology for the organisation.
- Co-ordination of efforts: providing information across projects, building infrastructure for the organisation .
- Changing the culture.
- Asset management.
- Managing external vendor relationship.
- Business strategy synchronisation.
- Evaluation for appropriateness for classes of projects.
- Networking, consultation.
- Evaluation and research.
- Standards, guidelines, regarding design review, reuse process etc.
- Promotion and selling OO to management.

5.3.3 Guidelines

5.3.3.1 Metrics

Approaching management purely with technical aspects will not succeed - one has to demonstrate improved productivity. Unfortunately there is not a lot of information on metrics for OO.[44]

"Metrics are fundamentally different for OO. The OTC should push for the development of such metrics and need to help drive this development by working with projects to understand and validate what types of data are important to collect" [44]

5.3.3.2 Management

The following guidelines are given:

- Getting management involved early .[45]
- Providing documented tools and techniques.[45]
- Mentoring should be hands on. [45]
- The transition to OO should not happen too fast.[88]
- The expectation of reusing OO in the first year[88] should not be created.
- One programming language is recommended. [88]
- Finding a champion that will own the vision is also recommended.[88]

5.3.4 Implementation

Various USA companies are currently busy with the following kind of activities in their OTC centres: education, OO hotlines, tool evaluations, monitoring project status etc. [45]

There seems to be 2 groups of OTCs: those focusing on education and technology transfer, and those focusing on infrastructure and providing reusable entities.

5.4 Project size

Ed Yourdon's comment in 1990 was that "A system composed of 100 000 lines of C++ is not to be sneezed at, but we don't have that much trouble developing 100 000 lines of COBOL today. The real test of OOP will come when systems of 1 to 10 million lines of code are developed". [24]

He proved to be right:

In 1994, Hartman [37] reported that most current OO methodologies and tools do not yet provide a suitable level of coverage and support for the construction of large-scale systems, including:

- The ability to drive the process from the business models and not from the requirements statement.
- Addressing usability aspects from the very beginning of object model development.
- Given the event driven loosely coupled nature of these systems, state transition needs to be widely supported.
- Contract definitions need to be addressed throughout the development.

Also, in 1995, Vayda's experience [59] in large scale projects was that "applying the OO approach in the industrial setting turned out to be a battlefield strewn with landmines in a number of surprising areas".

He provided the following characteristics that make large projects prone to problems:

- Information systems grow in a piecemeal manner over a long period of time. There is no model to explain how the various applications databases and platforms work together.
- Different groups helped develop the project and therefore have different understandings of the problem.
- The application has to be integrated with other applications.
- A platform change is part of the requirements.
- Data relationships are complex and databases are large.
- There are high performance requirements.
- The system requires friendly user interfaces.
- The solution must be scaleable.
- There is a lot of inertia and resistance to change.
- There are a lot of political issues.

The situation today can best be summarised by the Cutter Consortium's conclusions:

"Large companies are generally ahead of smaller companies in adopting OO. Large companies are likely to have CORBA and C++/Java while small companies are ahead in adopting OO databases, reflecting the stronger hold that database managers in large companies have on their organisations." [34]

5.5 The role of Quality and ISO9000

According to a survey amongst panellists[31] when asked whether there a relationship between the migration to OO and "process improvement" and the capability maturity model, it seems that the majority of panellists felt that "process is still more fundamental than which methodology you choose." It is felt that one should rather then use structured methods with a well-defined process, than use OO languages such as C++ and Smalltalk and hack. Organisations often begin using OO without having any software engineering process as a foundation. The benefits of having such as process can be found in the statement "Software development

processes map the abstract theories of the OO technique into concrete and repeatable actions"[28]

The next section explores the role of quality in the transition to OO.

5.5.1 Arguments for Quality and ISO9000

OO will only benefit the organisation if the correct processes are also in place. Although the NASA Software Engineering transition to OOT took 7 years, this is typical in any *ad hoc* environment

5.5.2 Against Quality and ISO9000

Korson[43] argues that the more mature a company is, the more reluctant the company is to move to OO because of the immature state of the process infrastructure for OO software development.

5.5.3 SEI level

The necessity of having an appropriate SEI CMM level[2] for success in OO is discussed in the following position papers:

According to Baer[23] it is not necessary to achieve some SEI level to move to OO, however he recommends moving to level 3 to maximise reuse.

Bernsen reasons that the SEI level can influence the ease and duration of the transition to OO. However, the transition can only be as orderly as the SEI level. On the other hand high SEI levels could cause difficulty as the company has set practised processes which may lead to inertia to change. He therefore feels that companies at level 1 should consider the transition to a higher level concurrent with moving to OO.

Bulman[24] believes that if a choice exists between shifting to OO and improving the SEI rating, the shift to OO should be done first. He reasons that as the organisation moves up the maturity scale it will be with the better methods. If an organisation chose to first move to a higher level on the SEI scale, and then changes to OO this would cause a temporary drop back down the scale because different parts of the organisation making the shift at different rates will result in inconsistency.

Korncoff's comment was that large scale reuse goes together with the SEI level.

In his article[89] Vayda confirms again the importance of ensuring a quality process, saying that OT requires the development processes required by the SEI (e.g. version control, metrics, inspections, etc) to be handled very well, regarding the following:

- consistent Analysis Design and documentation
- change management
- traceability through the system development life cycle

The author therefore strongly recommends, along with the adoption of OOT, the adoption of a process improvement program.

Baer [23] said that although lots of companies have small pockets of OT, when it comes to using it on a large scale, using objects requires understanding how they impact every facet of the system development life cycle which again enforces the drive towards quality.

According to Bulman[23], it is not recommended to change every aspect of the software process when making the transition to OO. "The need to understand the users of the software, and communicate with them in a way they understand does not change. This is the core of every development process and must dominate the devising of new development methods. It simply does not matter how well you solve a problem if it is not the one your customer wants solved."

A note of warning is also given: the development environment in any large organisation must provide support for both an OO development process and the previously used process. No organisation can commit suicide for the sake of a new "truth."

[2] The CMM (capability maturity model) was developed by the SEI (Software Engineering Institute) at Carnegie-Mellon University, and defines five levels of maturity in the software process of organisations.[80]

5.6 Configuration Management

According to Yourdon,[92] configuration management is a major issue in OO projects because of inheritance. Inheritance is one of the technical virtues of OO, but it can also be an Achilles heel.

It is also important because of iteration rapid prototyping in small multi-person projects. OO projects need good configuration management tools for version control, impact analysis, group ware issues etc.

5.7 Experience Reports

"Risk management is mandatory. We can either learn from others' mistakes or create our own to learn from" – Love [53]

"Ten times more people showed up for a recent Java conference than for the most recent object oriented conference. Java has gone mass market (because of objects). But mass market users ...have not read 10 years of JOOP or OOPSLA proceedings. Many are just beginning to make the mistakes that were well documented eight years ago"

5.7.1 BNR

At BNR the developers learned the following valuable lessons[59]:

- continuous integration is crucial to successful integration of applications and frameworks.
- a common design vocabulary is necessary. This is where patterns can be useful.
- cultural changes can be a significant obstacle. Learning to communicate requirements in terms of problems rather than solutions is important.
- high cohesion and coupling of objects can be used to one's advantage in organising the people as well as the software but care must be taken to ensure than the organisation mimics the architecture rather than vice versa.

BNR's problems were the following[59]:

- requirements were captured in a variety of formats and levels of detail - some too detailed, others too abstract.
- There was a lot of repetition of requirements
- the number of use cases made the task of analysis impossible.
- requirements were written without a common vocabulary
- designing class interfaces was difficult

5.7.2 Knowledge Systems Corporation

Reed Philip from Knowledge Systems Corporation[43] has helped companies with the transition to OO and found the following problems:

- Team sizes are often too large
- Management resists iteration and question the investment in refractoring classes that seem "to work as they are"
- Languages support a rate of change that is faster than an organisation can support – this causes programmers to add new features faster than the documentation team can document them
- Tools are still maturing
- Reuse is often not well understood and easily oversold. Most classes are not reusable outside the system they were developed for. Also, most companies lack the organisation and communication to support reuse.

5.7.3 OS/400 project

The OS/400 project described in Berg's article[12] found the following success factors:

- Team members became proficient in C++ using it for OO and not just a better C
- Having clear objectives helped
- Having experience in an object based background helped

- management consisted of former developers which therefore had the necessary technical background.
- individual teams were rewarded by schedule and not lines of code.
- frequent system builds turned quality into a reality rather than being an abstract concept
- training and mentoring
- proper inheritance was emphasised during training
- design patterns were useful during training
- CASE tools were mostly used to capture designs and the whiteboard for producing the design
- they used a performance engineering team
- C++ was found to be very effective. The flexibility of C++ however also lead to more options which increased complexity

The following problems also occurred:

- Code bloat due to the use of templates.
- Multiple inheritance was not very useful at implementation level.

5.7.4 NASA

At the NASA Software Engineering Laboratory (SEL)[51] using OO has decreased costs, shortened schedules and made the system more reliable. They found that 80% of the benefits of OO stemmed from abstraction and encapsulation, providing the most economic benefit for costs of projects, rather than inheritance and polymorphism and dynamic binding (although these were still helpful).

Balfour said that "inheritance and polymorphism may be the fun part of OO, but it's the more pedestrian parts that are giving us a lot of the benefits"

The description[86] of NASA Software Engineering Laboratory's 7 years of experience provided the following insight:

OO was introduced with the promise of reuse yielding benefits in the cost and reliability of software products. It was expected that OO would be more intuitive than the structured development traditionally used. Therefore it was expected that the cost of developing new code would also decrease.

During the 7 years that the study was done it was found that OO does promote reuse but some times neglects important issues such as run-time efficiency.

It was found that the use of OO is not as intuitive as expected, partly because the technique was new to an organisation with a mature structured development process. Also, it was felt that skilled designers are still needed to solve difficult domain specific problems. On the other hand OO was found to be the first technology that covered the entire development cycle which resulted in productivity improvements and decreased development cycle time. In this sense it was the most influential technology studied by the Software Engineering Laboratory.

5.7.5 MPR Teltech

MPR Teltech's experts[87] felt that the following actions lead to their OO success story:

- Educating the team and allowing for prototyping
- Using objects from analysis through to design
- Using an iterative approach to allow for parallel development activities.
- Using the right implementation language
- Developing on multiple platforms

5.7.6 DCE systems

Experience reports[8] detailing the blending of OO and DCE indicate that this has not been easy. The following problems had to be faced:

- C++'s objects are single process objects

- There were existing legacy C-based applications that needed to be reused in the short term

Regarding the lack of experience in the developers the following issues were stressed: mentoring, reuse, encapsulation and iteration.

They based their distribution approach on the following:

- Using proxy objects encapsulating distribution
- Using an OO DCE
- Using a consistent process model
- Using a consistent approach to testing
- Iterative development – many of the executables were built from third party and platform libraries. Finding out how they coexist in an application is vital.

5.7.7 *Caterpillar*

Caterpillar's transition when moving from COBOL and mainframe[7] lead to the following revelations:

In 1992 it was felt that there were sufficient OO tools commercially available to support the development of a number of small prototypes. The hands on experience with the prototypes would then permit the evaluation of OT.

The following problems had to be overcome:

- The majority of the staff consisted of mainframe COBOL developers with little or no PC experience.
- They had very little experience with iterative or rapid prototyping techniques.

It was decided to follow a pure object oriented approach that would focus on reuse, encapsulation, etc. Since there were no OO COBOL tools available, Smalltalk was chosen since it was perceived to be easier for COBOL programmers to learn. CRC cards were used since this was relatively simple to learn.

The following problems occurred:

- They had a difficulty in stabilising the design due to so many CRC cards and not having automated tools. It was felt that supporting tools therefore needed richer functionality.
- It was not clear how to handle persistent data within an object oriented application.
- Without automated tools the current process was not self-documenting.
- With prototyping it was not clear when to stop iterating.

5.7.8 *The Eagle project*

The Eagle project undertaken by Andersen Consulting[37] highlights the following problems:

- a clear requirements specification does not always exist, as sometimes assumed by the specific methodology being used.
- typical methodologies often do not address the usability of the system.

Because they used framework-based development, 80% of the entire system code resided in application architecture, middleware, and operating system layers. Only 20% of the code would be in the application specific layer. This breakdown translated into a significant maintenance saving over time. Eagle's concept of component-based solutions therefore focused not on the object level but at the component level.

5.7.9 *Hewitt*

Harrison[35] describes how in 1989 they found object instances to provide a natural way to model program constructs, and to capture complex relationships between different aspects of a software system.

They felt that the object paradigm could be efficiently implemented on standard hardware and software and that it provided some degree of extensibility without requiring major modifications to the existing implementation.

According to Tim Hilgenberg,[43] they therefore decided to introduce OT even though there were

not many tools available in 1991. At the time they experienced the following problems:

- The team had to be educated – they solved this using formal training and lots of practice.
- They struggled with developing a technical platform for development in the runtime system without a tool to help.
- They had to manage reuse in a large project – to do this they created a repository group within the project organisation.

5.7.10 Chembench

At Chembench[69] designers chose the evolutionary approach and used a wrapper to include the existing software.

Their advice was ensuring consistency so that the conventions and rules a programmer learns with one class is adhered to by all classes, thus making it easier to use new parts of the system.

5.7.11 Digital Consulting

Digital Consulting's OO Pilot Program[49] found the following to be success factors:

- management support and understanding of the technology
- emphasising OO analysis and design and not just coding from the start
- providing mentoring through consultants
- spaced and timely training geared to actual projects, for both management and technical staff.
- concrete milestones
- having a formalised OO software development process
- regular technical reviews
- having a stable and mature organisational structure that can undertake the technological change
- Implementing a change in culture, where the organisation has incentives for technical staff to learn and adopt new technologies
- Carefully selecting and planning the first few pilot projects
- Having different project structures, roles and responsibilities to adapt to the OO paradigm

5.7.12 Hewlett-Packard

In 1995 at Hewlett-Packard[54] OO had been used throughout the company for up to five years in divisions including telecommunication, medical, measurement, network and printer markets, and manufacturing. The common set of drivers were:

- Faster development making it possible to produce derivative products quickly
- dealing with complexity: using OO helped with abstraction
- software development became more manageable
- managing larger teams were made possible
- legacy systems could be evolved

Compared to the study in HP in 1990, objects were now being applied with more confidence and more success. Projects were getting bigger in team size. Since the 1990s study there had been only 1 large scale failure. Also in 1990, reuse meant class libraries only, whereas in 1995 several divisions were using OO to successfully develop domain specific frameworks for product families.

The lessons they learnt were:

- investing in a training program that includes classes, pilot projects, mentors, and reviews.
- Planning evolutionary development cycles based on OO models.
- staffing according to the natural distribution of work that occurs as development progresses through analysis and design. (The high level architecture is best done by a small dedicated team.)

- measuring progress by completion of OO analysis and design models. This includes maintaining a metrics database
- introducing processes to enhance communication of OO analysis and design models across teams.
- adapting OO to meet project requirements.
- recapturing the design of legacy systems in OO analysis and design models.
- regarding tools, the CASE tool situation was not satisfactory. Most users regarded CASE tools to be very important.
- Current methods did not address the problems of how to scale with the size of the problem.

De Champeaux's 1992 study[20], also at HP, provided the following insight:

- unlike in structured methods where there is a discontinuity between analysis models and design specifications, (because data flow and process models become control flow and interface descriptions) in OO the analysis model became a powerful framework on which the design was built.
- the defect rate with C++ code was 50 % less than that for C code on prior projects.
- a complete requirements document was crucial
- regarding reuse it was found best to separate the various phases, since some parts were more reusable than others.

6. Technical Issues

6.1 Development Issues

6.1.1 Problems

Aksit[2] identified several technical problems in 1992:

Problem 1: Problem domain structure: Identification of problem domain structures is difficult and often underlying theories of large systems are not completely understood. It is therefore difficult to define reusable hierarchies.

Problem 2: Excessive domain objects: Designers often introduce many objects even though only a few of these objects are relevant to the problem at hand.

Problem 3: Early decomposition: Where the software engineer does not identity subsystems of the bigger system before starting with the identification of objects, the project easily becomes unmanageable due to a large number of objects being identified.

Problem 4: Subsystem object distinction: Difficulties arise due to the distinction of subsystems (which Booch calls class categories) from objects.

Problem 5: Commonality versus partitioning: Subsystems are assigned to different engineers making the design of inheritance hierarchies very difficult.

Problem 6: Subsystem identification: Specification of the interaction between subsystems can become very complex in large software systems.

Problem 7: Sharing behaviour with state: Instances store states whereas classes behave as templates, defining the common features of their instances. However, for some applications it may be desirable that the state shared by instance objects affects their operations defined at the class level. At the time, this could not be expressed in the available OO models.

Problem 8: Inheritance versus state: Most OO models did not cater for the integration of states with inheritance.

Problem 9: Multiple views: There was a need for multiple views on an object where not all the operations provided by an object were necessarily of interest to other objects.

Problem 10: Language Database integration: Most OO methods did not address database related issues such as persistent data structures, transactions and queries in software development.

Problem 11: Co-ordinated behaviour: Object interaction is based on the sending of messages between objects. The authors considered this model to be unsatisfactory since it only involved two partner objects at a time.

The problems listed above were identified on the type of projects given in Table 6.1:

Table 6.1

Problem	1	2	3	4	5	6	7	8	9	10	11
Administration system	✓		✓				✓		✓	✓	✓
Network database		✓								✓	
Chemical process control system		✓									✓
Mechatronic modelling system		✓							✓		✓
Intelligent tutoring system	✓						✓		✓	✓	✓
Concurrent processing								✓			
Distributed office system	✓						✓		✓	✓	✓
Parser generator											
Distributed system design	✓			✓		✓					✓
Temperature control system					✓						
Intelligent mail system									✓	✓	

Hϕydalsvik[40] evaluated OO analysis (OOA) and found that OOA had not yet delivered what it claimed to do:

- OOA does not meet the full needs of the analysis phase. Many methods assume that the requirements have been met before the analysis starts.

- the transition to design is not always as easy as promised.
- objects are not sufficient for an adequate representation of real world concepts.
- business rules are not easy to capture within one specific class because they tend to be global of nature
- Dynamics are not captured. Some OO approaches have tried to fix this e.g. use cases and scenarios. The dynamic modelling proposed by these approaches are bottom up rather than top down.
- at the time of writing the article (1993) most languages and methodologies for OOA did not support verification or validation

Martin [58] reports that OO has solved the problems inherent to structured programming, but that although the old problems are now forgotten, there are new problems:

- Successful OO programming requires more theory and knowledge than structured programming
- OO programming offers a few pitfalls that are not present in traditional programming such as through the incorrect use of inheritance.
- OO programming in connection with strong typing still has problems regarding parallel type hierarchies
- There are numerous minor issues that have not been settled yet in OO. These include whether classes should be the only module concept available, whether subroutines should be allowed to live outside classes and how to handle multiple inheritance.
- OO is new which means that lots of standard recipes (design patterns) have not been developed yet.

6.1.2 Guidelines

6.1.2.1 Implementation

According to Vaydal[ ], knowing only language syntax and semantics is not sufficient.

- The cost of using certain language features should be understood, for example the fact that templates cause an expansion in code size.
- It is important to understand polymorphism well before using it. Problems exist regarding documenting behaviour of polymorphic inheritance hierarchies, handling polymorphic types stored in multiple containers, etc.
- One should know the compiler and avoid compiler specific features. Knowing the compiler will also help in the error prone areas e.g. memory management, copy construction, etc.

6.1.2.2 Selecting the OO technique

When selecting an OO technique, Fayad[29] recommends selecting a fully object oriented technique that covers the whole software life cycle.

6.1.2.3 Finding the right objects

Coad[17] provides several guidelines to finding the right objects, such as:

- Investigating the topic – classes can be identified by using a book on the topic to see how the author has organised it.
- Breadth – this includes going beyond the domain of the problem statement. This will also help with reuse.
- He recommends using a team approach and not the traditional "first analyst then designer then programmer" approach.

6.1.2.4 Object size

The authors[70] reason that the size of the objects are important:

Very small objects are small enough to be reusable since they can only be interpreted one way. Very large objects are trusted due to their robustness and are therefore also reused.

It is the other size objects that are not being reused, as they are open to the wrong assumptions being made about the functionality. Unfortunately most OO methodologies concentrate on medium size objects as examples in their books.

6.1.2.5 Problem oriented analysis

According to Hϕydalsvik,[40] OOA should become problem oriented (i.e. concerned not only with what a modelling language should be able to express but also how) rather than target oriented. (i.e. a solution oriented approach)

6.1.2.6 Prototypping

Korson[43] warns against never ending prototyping.

6.2 Some Hybrid techniques

6.2.1 A structured analysis phase

Kerth[42] suggests a structured approach to object oriented design:

"In fact, bringing object-oriented concepts such as inheritance into the analysis phase, might cause serious errors in one's analysis"

6.2.2 Transformation of data flow models to move from SA to OOD

In Alabiso's article,[1] a strategy is proposed to migrate from the decomposition of a system analysis performed according to structured analysis, to the design of the same system according to object-oriented techniques. The authors feel that this method is not only possible, it is also useful, arguing that:

- Structured analysis techniques such as data flow diagrams have amply demonstrated their value in expressing the specifications of the functional requirements of a system.
- Data flow lines can be dubbed object flow lines and names entered in the data dictionary truly correspond to object names.

A concern about this proposal is that structured analysis techniques do not cater for inheritance. However, one can argue that the process of organising classes in a hierarchical fashion is truly a design time task anyway.

In De Champeaux's article,[21] arguments against the above include the notion that structured analysis characterises processes first and subsequently derives the members in the data dictionary. This then precludes the identification of classes, does not exploit inheritance and prevents encapsulation.

6.3 Methodologies

When asked about the importance of methodologies (sometimes referred to as methods) in software development today, Stroustrup[66] replied:

"For larger projects, the rules and processes we call methods are necessary, for smaller projects less rigorous approaches are often preferable. Methods are too often used in attempts to compensate for lack of direction, for lack of a conceptual framework for the system being built, and a lack of concepts in the programming used. A method is not a substitute for thinking and understanding. Methods should be applied flexibly enough to accommodate the varying talents, tastes, and weaknesses of a diverse manager, designer, and programmer population."

Methodologies should therefore be used correctly to be of value in the making of OO systems.

6.4 CASE tools

6.4.1 Arguments for CASE

"While vice-grips may be used to drive in a nail, a craftsman will always use a hammer!" - Baker[6]

From the comments of various authors one can see what an important role CASE tools have to play:

Coad[25] sees CASE and methods as the necessary tools to communicate expertise to get the job done.

However, Grady Booch warns that unfortunately CASE tools allow poor designers to produce bad designs much more quickly. He has also found that there is a threshold were CASE tools get in the way but that in certain classes of problems a certain level of notation is not only necessary but highly desirable to raise developers to higher levels of abstraction. Notations are necessary to express what cannot be express textually. Designs also need to be preserved for the future.

Meyers said that good tools help to maximise the productivity of developers of any kind of software development.[61]

Finally, Narayanaswamy[63] forecast that OO methods will see increased usage in the future, but that without computerised support in the form of CASE tools, it will be impossible to contemplate OO technology in large projects.

6.4.2 What is there now

This can be found in Artsy's[5] position paper on the question whether object-oriented CASE tools are ready for prime time.

"...In recent years CASE tools have rapidly advanced from mere drawing packages to overall modelling systems, offering faster and nicer graphical drawing capabilities, rule checking and enactment, more slick interfaces, and even code or database generation. Today's leading or new OO CASE tools, including OOA&D tools, are good at helping the designer to model the application under development, and at saving manual work by generating portions the application - thus paying back for their cost by cutting backlogs and sparing development resources."

6.4.3 Why there are problems

According to Hazeltine[43] some groups take too long to decide which tool to use and when they do decide a new tool comes out. The technology itself is still more important.

"The continual introduction of new products makes it easy to develop a love/hate relationship with tools", said Shan[84]. The author found that while it is important to be quick to adopt new tools that can help speed development and improve quality, others must be avoided because of additional training times, unnecessary complexities, and inappropriateness to the job at hand.

There are certain technological trade-offs that cause major tension in the way CASE tools work:[63]

- Deep semantics requires a formal notation
- Large OO models require viewing and navigation, but keeping consistency across views is complex.
- The front end of CASE tools need to be flexible but the back end has to be completely compatibility with execution engines which causes complexity.

6.4.4 What lacks at present

6.4.4.1 Large systems

With large systems there are the following problems with CASE tools:[46]

- A lack of holistic view (to expect detail but also see the overall structure)
- how to track objects that are out of focus
- how to hide uninteresting information
- how to match dynamic runtime behaviour to static descriptions
- how to find patterns

6.4.4.2 Opportunities in the market

According to Malan[54] CASE tools at the high end are not intuitive to use, whereas tools at the low end provide minimal support. There are therefore definitely opportunities for

improvements in the CASE tool market regarding:

- flexibility
- integration
- team support
- support for reverse engineering
- reverse traceability
- support for reuse

6.4.4.3 Standardisation

Standardisation of tool integration is still not complete which means that CASE tools might not be ready for full time deployment and can therefore cause more problems in large-scale projects. [30]

6.4.4.4 Life cycle

Most CASE tools do not cover the full life cycle.[11]

6.4.4.5 Support

It is felt that there is a lack in on-line support to help the user. [11]

6.4.4.6 Other problems

Narayanaswamy[63] mentions the following:

- Maturity/robustness of tools
- Adaptability to real project situations
- Vertical versus horizontal tool support: does the tool cover the entire software development process, and does the tool provide integration with other tools?
- Inadequate simulation capabilities
- No metrics
- Interoperability between CASE tools and the development environment
- Configuration management at the model and other levels
- No support for incorporation of non-formal knowledge

6.4.4.7 A backwards evolution

According to Pircher,[75] software development paradigms have traditionally evolved backwards, starting with implementation and moving back towards the beginning of the development lifecycle.

Unfortunately tool vendors also followed this approach, concentrating on the detailed design including the implementation cycle, forgetting that the major benefits of object orientation are achieved by making sure the right system, based on requirements, is built.

6.4.5 What CASE tools should have

Yourdon provides his guidelines[92] about what a CASE tool should have:

- support for graphical notation
- providing the ability to hide and reveal layers of a model at request of the user
- having the ability to show different message threads
- browsing capabilities
- error checking for completeness, consistence etc.
- interface to other CASE tools, repositories, code generators etc
- groupware support
- the need for an inexpensive tool to play with so that SEI level 1 organisations can buy tools and throw them away afterwards

6.4.5.1 Unrealistic requirements

CASE tools often have unrealistic process requirements regarding the order of specification of a diagram. More flexible architectures such as whiteboards should be available.[63]

6.4.5.2 Feedback

Early feedback is necessary for validation. [63]

6.4.5.3 Code generation

The more code is generated automatically, the more programmers can address the real design issues.[61]

6.4.5.4 Other requirements

The following additional requirements were given by various authors:

- interoperability without the use of centralised repositories[63]
- standards for models and libraries[63]
- legacy system integration into models [63]

6.4.5.5 Visual and audio techniques

Laffra[46] requires the following to help with the understanding and debugging of OO systems:

- a query mechanisms for filtering information
- three dimensional views
- customisation of views
- abstractions on data and views
- catering for novice users versus novice programmers
- multimedia
- mappings between OO and visual techniques
- combining different paradigms
- openness and extensibility[75], including support for local standards e.g. language standards, support for own or third party tool integration and the integration with the local Configuration Management system which is often the backbone of the full life cycle environment.
- Repository support [75]
- Support for ad-hoc queries from the user. [75]
- Support for large system consistency. Building large systems requires tools to support consistency checks between different model components. [75]
- Support for traceability is necessary since one of the central promises of OO is building systems that better match real world requirements. [75]
- The ability to build, share and reuse components across projects should be provided.[78]
- Support for a multi tiered distributed open repository[78]
- specific requirements for compilers and runtime systems
- cognitive aspects

6.4.6 Selection of CASE tools

6.4.6.1 How to select

In many cases the tools themselves contribute to the complexity of the system.[6] The thinking of "one size fits all" indicates an immaturity in issues relating to CASE tool selection. CASE tools are ready for the kind of systems being developed but they should be selected correctly.

Critical factors for success when selecting CASE tools are:[48]

- The tool must support a methodology that maps well to the problem domain. Baker[6] confirms this, reasoning that the tool must provide an expressive notation as many too

today are geared towards a limited problem domain.

- It must fit well into the larger development context, e.g. the chosen implementation language. Also, the design and code must be tightly integrated so that modifications can easily be made at design level.
- The tool must provide for early validation of designs, i.e. before code generation starts. "At a minimum this means defining all legal messages to flow into each active object, and running test scenarios"
- It must provide good modelling support for object behaviour so that it is possible to display both message traffic and state-transition activity of active objects.
- It must allow integration of diverse class libraries or packages
- The tool should provide adequate support at the design level for exception handling
- No CASE tool can be used without a learning curve. The development organisation should allow for training and experimentation with the tool, and should invest in tailoring the tool for the particular environment

Adding Baker's[6] selection guidelines gives:

- The toolset should support the entry of source as graphics since a picture is still worth a thousand words.
- The notation to describe the system should be the same across all phases of development to ensure that all the teams speak the same language
- A family of interoperable tools must be integrated into an integrated environment so that the integration of requirements analysis, source management, document support, etc. can be seamless
- The tool should provide the ability to define a system architecture that reflects the organisation of the development teams.
- Page-Jones[71] recommends performing a hands-on evaluation of a CASE tool before committing to it.
- According to Coad[25] one should use the CASE tools whose vendors also use them.

6.4.6.2 CRC cards

The use of CRC cards provides a physical understanding of objects and prepares users to understand the vocabulary and details of particular languages. [10]

It is also cheap, portable readily available and familiar as everything is written on a 4" x 6" index card.

6.4.7 The Aaron Project

The principal goal of the Aaron project[64] was to ascertain how CASE tools can effectively support the managed development of quality software.

"OO CASE tools are very useful infrastructure for the acceptance of object oriented methodologies into mainstream software development. Many existing OO CASE tools are, however, simply adaptations of non-OO CASE tools to support OO diagram notations. The Aaron Project at the University of Technology, Sydney, is investigating the design of next-generation CASE tools to ensure they fully support OO methodologies. The project's current focus is the usability of CASE tools and their integration into the development life cycle. "

6.5 Object Oriented Programming Languages (OOPLs)

According to Martin[57] "things such as support for roles, constraints, change in state, etc. may not be missing from OO as a whole but from the implementation of some of the OOPLs."

6.5.1 The pragmatism

Yourdon[92] provides the following reasoning regarding languages: Full blown OO is still the most desirable way of supporting OO. However, less than an OOPL is a reality, and can still be effective if combined with style guides, discipline and perseverance.

6.5.2 *Which language*

The problem with pure OO languages is that programmers are forced to choose between purity and performance. All computation is performed by sending messages between objects, but these languages cause high call frequencies, which interfere with good performance.[14]

Yourdon[92] selected the following OO language evaluation issues:

- Class, object
- Generalisation specialisation, including single versus multiple inheritance, conflict resolution and "name conflict resolution"
- Whole part structures
- Attributes, which includes support for instance connections, visibility and constraints on attributes
- Services, including support for message connections, visibility, and dynamic binding (the ability of an application to select a particular service at run time)

6.5.3 *C to C++*

"Because C++ is considered the successor to C, C programmers wanting the benefits of the object oriented paradigm look to C++ as a logical step toward object oriented programming" – Reed. [76]

There are two methods of performing this transition: one view promotes an incremental movement towards the new language and the paradigm, the other is more radical, prescribing adoption of both the new paradigm and the language at the onset. [76]

Barbara Moo's discussion[76] on comparing these approaches provides the following insight: it is much harder to convince a sceptical development manager to try a new technology if they have to start off with a large investment before seeing any return. C++ allows for the "learn while doing" approach, so that one can capitalise on the large investment already in existence in C knowledge. Then one can gradually evolve systems and people to use the data abstraction and OO facilities of the language. This provides a lower risk and provides time for local experts to evolve.

Stroustrup[66] recommends the following approach: "I encourage an approach based on an emphasis on strong static typing and abstraction techniques. I consider a heavy early emphasis on C or on class hierarchies problematic. There is a lot of mileage to be had out of strong static type checking and out of concrete and abstract classes before heading into the trickier parts of the common subset of C and C++ or the trickier parts of object-oriented programming."

6.6 Reuse

"Sharing is one of the primary benefits in object programming." – Shan[84].

6.6.1 *Problems*

Designing for reuse is difficult for the following reasons:[33]

- Over-generalisation leads to unbearably high infrastructure costs
- The inclusion of application specific ideas in a library intended for general use leads to restrictions and delay in subsequent projects

The successful adoption of reuse may involve major cultural changes and high levels of investment: [33]

- Organisations have to change their reward structure, with reuse specialists being rewarded quite differently from developers and developers being rewarded for speed and quality more explicitly.
- Key personnel will have to be taken away from the maintenance of important legacy systems in order to bring sufficient business knowledge to new OO developments.
- Managers will have to be re-educated in the new approach.

At Boeing[45] there are also still the following problems:

- Continued rewards for creation of parts rather than the use of parts (reuse)
- Tools for effectively supporting reuse are not yet available.

Vayda[89] distinguishes between organisational barriers and technical barriers to achieving reuse.
Organisational barriers include the transformation of the organisation mindset to include reuse, developing resources to champion reuse, and developing reward structures.
Technical barriers include the difficulty of producing truly general reusable components, documenting and distributing the components, finding the right components, handling changes to the libraries and namespace conflicts when integrating libraries from multiple vendors.

6.6.2 Guidelines

Korson's[43][44] recommendations for reuse are:

- Setting up a vision for an architecture based on business goals
- Making sure the goals are realistic,
- Knowing what granularity of reuse is needed
- Managing reusable assets
- Having proper tools to retrieve and store components
- Addressing liability issues such as copyright and infringement
- Providing rewards for developers submitting reusable classes to the class library
- Providing rewards for developers reusing classes
- Implementing good source control

6.6.2.1 Library management

The Ovum report[70] highlights the role of management, arguing that achieving reuse lies more in the hands of management and the organisation than in a particular technology.

Page-Jones[71] also recommends planning to have a robust, well managed library, with a librarian and consultants.

However, Korson[43] warns against so-called "Libraries of sofabeds" which are classes that have a large reuse potential but that are not optimal for any given application.

The development of libraries are therefore seemingly an important issue in the attempt to achieve reuse.

6.6.2.1.1 Maintaining the library

Page-Jones[71] provides advice, saying that formal policies for entering, storing, retrieving and removing library holdings have to be established.

When entering classes, caution needs to be taken to avoid classes missing methods, classes overlapping in functionality, classes that were not tested, several versions of the same class, classes without documentation, etc.

He recommends appointing a librarian to oversee the activities of entering, storing, retrieving and removing code. This librarian could be a single person or a small team. He also mentions the use of a library consultant to assist project teams in their reuse of classes.

6.7 Reengineering legacy systems

"While the advantages of using object-oriented design paradigm when embarking on the design of a new system have been established, the ramifications of using these techniques in the redesign of an existing system are less well-known" [69]

Dietrich[26] defines legacy systems as systems that evolved over many years and that are considered irreplaceable either because re-implementing would be too expensive, or because they are trusted by users. Because of their age, such systems are likely to have been implemented in a conventional procedural language with limited use of abstraction and encapsulation. The

lack of abstraction complicates adding new applications and the lack of encapsulation impedes modifying the system itself.

To understand legacy systems better, one can refer to Korncoff's[23] common characteristics:

- mainframe hosting
- layers of patch code
- fragility to change
- providing mission critical services

6.7.1 The importance of reengineering legacy systems

The significance of working with legacy systems can be found in the comments of the following authors:

According to Baer[23] the ability to identify successful strategies for working with these systems will determine the pace at which large organisations can move to object technology.

Jacobson[41] promotes the modernising of old systems in a gradual way as it is often unrealistic to replace an old system with a completely new system since changes require too much resources.

Graham[33] found the following scenarios in which an object oriented system should inter-operate with existing non-object oriented systems:

- The evolutionary migration of an existing system to a future OO implementation where parts of the old system will remain temporarily in use
- The evolution of important systems that are too large and complex to rewrite and where part or all of the old system may continue to exist indefinitely.
- Highly specialised or optimised routines, embedded expert systems and hardware specific software
- The continued use of existing relational databases
- The construction of graphical front ends to existing systems
- Integrating with existing systems across local area networks

Dietrich[26] found the following scenarios:

- The legacy satisfies most of the needs of the user but a better interface is needed to extend the system
- The legacy has code that can be reused.

6.7.2 Method of reengineering

Vayda[89] recommends four strategies for reengineering:

6.7.2.1 Incremental Reengineering

6.7.2.2 Database conversion

Controversy exists on this issue. According to Korncoff[23] legacy databases should be left intact.

6.7.2.3 Migration strategies

This includes running parallel systems until the new system is in place.

6.7.2.4 Wrappers

Numerous authors have mentioned this technique as a solution. However, several problems exist[26]:

- Interlanguage communication: since the wrapper depends on data that is internal to the legacy
- Garbage collection: if the legacy does its own garbage collection, it must be prevented from collecting data that is referenced by the wrapper. One solution is to ensure that for every pointer in the wrapper to legacy data, there is a corresponding non-garbage pointer referring to the same data.

- Memory compaction: if the wrapper has pointers to the legacy data and the legacy does compaction, the legacy may move referenced data without updating the wrappers' pointers. This can be fixed by having pointers in the wrapper referring indirectly to legacy data through a table of pointers in the legacy. Alternatively the wrapper should not use pointers to legacy data

- Object lifetime synchronisation: when de-allocating an object in the wrapper, the corresponding legacy object should also be freed or made available for garbage collection.

- Cross-system consistency: there should be cross system invariants that take interrupts into consideration.

- Increased orthogonality: the OO system should present orthogonal classes. Lack of orthogonal classes in the legacy system makes this difficult because wrapper implementers must now understand all of the undesired interactions in the legacy.

6.7.2.5 Other techniques

Korncoff[23] considers an OO client with legacy functionality acting as a server.

Jacobson[41] feels that the subject of reengineering is too focused on tools. Despite their importance they are not sufficient. He recommends trying to incorporate the reengineering as a part of the development process and not as a substitute for it.

6.8 Metrics

6.8.1 The importance of metrics

The importance of metrics can be found in the comments of various authors:

"Metrics provide a means of measuring process quality and identifying potential bottlenecks" [28]

"A business case for a CEO to make intelligent decisions to support the transformation can only be done if there is a set of metrics for cost and benefits. There has been little work on the quantification of the benefits."[43]

"In order for OOD to fulfil its promise in moving software development and maintenance from the current craft environment into something more closely resembling conventional engineering it will require measures or metrics of the process. While software metrics is a generally desirable feature in the software management functions of project planning and project evaluation, they are of especial importance with a new technology such as OO. This is due to the need to train current and new software engineers in generally accepted OO principles." [15]

Metrics that provide measures of the size and of the complexity of a software system can be used to aid management in the following:[15]

- Estimating cost and schedules of future projects
- Evaluating the productivity impacts of new tools and techniques
- Establishing productivity trends over time
- Improving software quality
- Forecasting future staffing needs
- Reducing future maintenance requirements

6.8.2 Inadequacy of existing metrics

Traditional software metrics do not provide for measuring OO concepts such as classes, inheritance, encapsulation etc. [15]

Moreau [62] found that very little research has been done towards analytically measuring and quantifying the advantages of OOD. It seems that existing metrics can be used only within a particular method within an object. The number of lines of code in an object is not a good indicator of development complexity, since only a small part of that code is likely to be unique to an object. (inheritance provides for code reuse among various types of objects.)

6.8.3 Guidelines

6.8.3.1 According to Moreau

Moreau[62] proposed the following metrics, which can indicate the complexity of a program:

- Message vocabulary size indicating the number of unique messages sent by the current object
- Inheritance complexity
- Message domain size indicating the number of discreet procedures within the object that manipulates its state

6.8.3.2 Coad / Yourdon

Coad and Yourdon use the following criteria: [92]

- Coupling
- Cohesion
- Clarity design
- Hierarchy and factoring guidelines
- Keeping objects and classes simple
- Keeping message protocols simple
- Keeping services simple
- Minimising volatility of design
- Minimising overall system size
- Ability to evaluate by scenario
- Evaluation of critical success factors
- Recognised elegance in the design

6.8.3.3 Chidamber

Chidamber[15] developed a set of language independent metrics that addresses the fundamental elements of object oriented design as outlined by Booch.

Metric 1: Weighted methods per class (WMC)

The number of methods and the complexity of methods involved is an indicator of how much time and effort is required to develop and maintain the object. Objects with large numbers of methods are likely to be more application specific, limiting the possibility of reuse.

Metric 2: Depth of inheritance tree (DIT)

The deeper the class is in the hierarchy, the greater the number of methods it is likely to inherit, making it more complex.

Metric 3: Number of children (NOC)

Number of immediate subclasses subordinated to a class in the class hierarchy. Generally it is better to have depth rather than breadth in the class hierarchy since it promotes reuse of the methods through inheritance The number of children gives an idea of the potential influence a class has on the design. If a class has a large number of children, it may require more testing of the methods in that class.

Metric 4: Coupling between objects (CBO)

Excessive coupling between objects outside of the inheritance hierarchy is detrimental to modular design and prevents reuse.

Metric 5: Response for a class (RFC)

If a large number of methods can be invoked in response to a message the testing and debugging of the object becomes more complicated.

Metric 6: Lack of cohesion in methods (LCOM)

Cohesiveness of methods within a class is desirable since it promotes encapsulation of objects. Lack of cohesion implies that classes should be split up into subclasses.

These 6 metrics are related to Booch's elements of OOD in Table 6.2:

Table 6.2

Metric	Object Definition	Object Attributes	Object Communication
WMC	✓	✓	
DIT	✓		
NOC	✓		
RFC		✓	✓
CBO			✓
LCOM		✓	

6.8.3.4 According to Li

The authors [50] proposed a metric for data abstraction coupling, a class interface increment metric to measure the size of the interface of the class and a metric to describe a software component.

They proposed that if metrics measured in the design phase are to provide useful information about software maintenance, it should be possible to predict the maintenance effort of a class from its design metric characteristics. Their study showed a method for doing this so that the maintenance effort can be measured.

6.8.3.5 According to Fraser

The following metrics for success were found[31], measuring not only technical progress but more specifically measuring the adoption of OO within the organisation:

- awareness: the percentage of new projects using OO, the number of adoptions, increased management support, a shift in vocabulary and satisfaction of the developers
- change: ease or speed of change
- communication
- methodology: this can be measured as completeness of the final project as it meets original requirements
- reuse
- team composition: productivity is used as indicator
- team dynamics: amount of rework required, people satisfaction, team participation, time spent in meetings
- technical training: number of programmers completing a curriculum
- tools: ease of maintenance, improvement in productivity

6.8.3.6 Using Z

The authors[85] propose using Z to measure complexity:

Many software metrics for procedural languages have been developed. Also, there have been a few complexity studies catering for OO systems, including those by Chidamber[15] and Moreau[62]. In Shih's article[85] the author proposes a highly mathematical set of metrics using the Z specification language, meant specifically for the inheritance hierarchy.

6.9 Testing

6.9.1 Importance

"Extensive testing and verification and validation is of utmost importance" - Fayad[28]

From Arnold's article[4] it is clear that testing is often the last item on the agenda and therefore is neglected the most:

"In the real world of large projects involving legacy systems, non-OO interfaces, non-infinite resources, testing is thrown back into the trade-off world along with all the other trade-offs involved in engineering and business."

6.9.1.1 Problems

Arnold found the following difficulties in testing objects:

- non thread safe C++ libraries
- non thread safe memory management
- lack of multi platform multithread debuggers
- different exception models

6.9.2 Guidelines

He includes the following aspects in testing objects:

- behaviour
- good citizenship which defines how well the objects coexist with others in the application
- consistency
- processes

Regarding the levels of testing, Arnold recommends the following: unit level testing, subsystem level, process level, domain level and cross-domain level (e.g. PC to UNIX).

His further recommendations include:

- using code analysis tools to ease the paper review process
- constructing a development environment that encourages and facilitates consistent testing project wide
- using self-instrumenting tools to assist in coverage, complexity and memory leak detection for multithreaded and distributed client server applications

Lastly, Graham[33] also recommends re-testing all subclasses when modifying a super-class and similarly, when altering a subclass, re-testing the inherited operations as well as the new ones.

7. The future

7.1 Is object technology still emerging?

"Given the close match between the advantages offered by OO and the current trends in business management, it's disappointing that OT has not gained more significant support among industrial software developers."[73] The author found the following reasons:

- No uniform software support for OO
- The existence of hybrid languages which are languages that add OO constructs to a procedural foundation e.g. C++, object Pascal, Ada)
- the number of pure OO languages that are limiting in terms of the platforms supported and the number of compilers available
- regarding databases, moving to a new system is time consuming even when good tools are available
- lack of standardisation
- lack of security capabilities
- no models for costing of OO programs. There are some metrics available but these are thought to be incorrect measures for OO.
- the technological problems mentioned so far may even prove insignificant compared to the

human factors issues that deter industry from adopting object technology

The characteristics of an emerging technology according to Ken Orr, (from the author Cockburn: The impact of object orientation on application developmen. 1993) are that there is more written about it than known about it, there are more people selling it than using it, and the vendors are making more money from education than from selling the tools". These still seem to apply to OO.[73]

It is also not clear how many people who think they are doing OO programming actually do that. It seems to be a small percentage and has not become the productive medium for the majority of programmers and designers in the field, says Orr.

"The problem is not the technology - it could be any technology.., the problem is cultural, its organisational, its people focused".

7.2 What is next for OO

ACM has convened an Industry Advisory Board sponsored by IBM to discuss future applications of Object Technology in industrial settings. The panel included experts from industry and academia to answer questions on the progress of OO and to look ahead to what we can expect in the future. [73]

7.2.1 Questions to be answered

From SEL's experience it seems that the initial experience with OO was successful but full OO throughout the software life cycle remains a goal of the future. The successes achieved this far took long to achieve.[51]

These questions were formulated at the time of writing but are still applicable:

- How radical is a shift to OO
- What is the likely time frame for OO insertion? Can it be accelerated?
- Would an organisation beginning OO now move faster?
- What factors influence the speed of OO insertion?
- How does one start introducing OO
- How should the risks and costs associated with moving to OO be dealt with?
- Is a revolutionary (shock therapy) approach preferable to a more gradualist evolutionary approach
- Is OO insertion best achieved top down or bottom up?

According to Pancake[73], the critical issues for OO research include:

- scalability of object implementation: resource management becomes difficult as OT applications grow to span multiple platforms. CORBA was the first step in dealing with this.
- interoperability: there is a need to connect dissimilar object systems - again CORBA is a start in that direction.
- tool and language support: Panellists agreed that commercial products are still a long way from the level of simplicity necessary to appeal to large number of users. Visual programming and graphical elements are there but do not really represent the basic object concept. Tool integration is a problem: the user still has to use several tools for design, implementation, documentation, etc.
- support for learning OT: the current style of teaching OO put too much emphasis on language and implementation. There is a need for large-scale examples.

7.2.2 Component Technology

Graham[32] commented on the lack of good books and seem to think that there is currently an "Object winter" analogous to the AI winter of the 80s. He found that some people think that object technology is dead and that it will be replaced by "Component Technology".

7.2.3 Distributed Applications

Various references have already been made to the 1997 report on the use of object technology at companies throughout the world. A section of this

report concentrates on companies use of OO middleware. The results were as follows: the Internet (33%), CORBA (29%), DCOM (12%), Netware (8%), other (6%) and lastly none (12%). (The Internet supports both CORBA and DCOM.) CORBA is therefore the dominant standard at this stage.

Many companies are still investigating these standards. Both CORBA and COM are still rapidly evolving. Also, since the survey was done, there has been various new developments, including COM+, RMI as well as various products that will incorporate CORBA.[35]

7.2.4 The role of patterns

Coplien[65] hopes that patterns will, like objects, become second nature in mature development environments.

7.2.5 New languages

In his article, Cummins[19] explores the features and capabilities of next generation object oriented languages. These are:

- Artificial Intelligence deals with complex problems in an effort to automate human capabilities. Objects provide a powerful base for representation of concepts. Together these approaches can provide solutions to very complex problems.
- An open object oriented paradigm

  The common base mechanism should support the various paradigms that in turn supports an application.
- Building systems

  In order to promote reuse, the concerns about how an implementation is documented and how much of it must be accessible, should be resolved. If too little information is given the user cannot use the object but too much can violate the developer's intellectual property rights.

7.2.6 Security and safety critical software

In his 1994 article Herndon[38] highlighted the need for standards for security in OO systems:

"The increased popularity of OO systems has also heightened the need for OO systems that operate securely, this includes access control, identification of users, auditing, administration and secure communication. The OMG have made available a white paper addressing security within CORBA. Issues to be sorted out include how to provide more privacy within a distributed object system. Security practitioners expect a more centralised access control component to be provided, and see distinct advantages in the object paradigm for doing enforcement. Vendors on the other hand see a more decentralised system that provides added flexibility without trying to solve the difficult problem of supporting general system interoperability at the level of a secure ORB. Much work has to be done to provide robust flexible and still assurable secure distributed object systems."

As mentioned before, since then we have already seen a lot of progress in the development of CORBA and its implementation in various organisations. Further development will take place, as systems become more complex.

According to Riehle [77] "there is a growing interest in the use of formal methods for safety critical software. That is designing a software specification so it can be proven to be correct."

He says that "the ability to support formal methods is one of the challenges for object oriented programming if it expects to be taken seriously in the real-time safety-critical marketplace.

7.2.7 Server objects

One can refer to various comments in favour of server objects: [83]

7.2.7.1 John Tibbetts

Having server objects is the end result of a long-growing trend to make servers chunkier and clients thinner. This will make it possible to think about client design and server design separately.

7.2.7.2 *Woolfrey*

The server is where objects provide the greatest advantage. In his experience the hardest business problems are shared database, multi-user, server-based applications. Objects will help manage this complexity.

7.2.7.3 *Tom Morgan*

From his experiences at Brooklyn Union Gas Company it seems that the greatest difficulty lies in the widespread use of server objects. There is a lot of semantics to be conveyed about the objects and what they do. Client objects are understandable since they are used everyday.

He also believes that the recent legitimisation of dynamic object oriented languages (referring to Java) will provide a strong support for moving towards server based objects.

7.2.7.4 *Phil Proudfoot*

Proudfoot believes that while objects have assured themselves a place on the lower risk client side of the client/server equation, they have yet to make significant inroads into the back office, server side of the same equation. He argues however that this situation is definitely changing for the following reasons:

- The important object languages (Smalltalk, C++) either currently support or have planned support for mainframe or host class server environments.
- Clients are reaching the boundary of what they can provide without changing the server application. These workstation applications are becoming overly complex and will soon represent the next legacy problem.

7.2.8 *The human factor*

When Coplien[65] was asked about the most advanced topics in computing today, his comment was the following, which again highlights the so-called "soft issues":

"I think the people problems are the most challenging, because those problems face one of the largest areas of ignorance in our discipline. ..Good human interaction is hard, and the individual actions that add up to communal accomplishment are complex. It's difficult to achieve optimised levels of human interaction, yet most organisations are barely competent at vesting architectural expertise, engaging customers, and valuing domain knowledge."

7.2.9 *Reengineering OO legacy systems*

Casais has recently highlighted a set of problems in a January 1998 article[13]. He writes that "experience shows that software developers have trouble imparting OO applications or components with the generality and adaptability needed for diverse and changing requirements". These problems include that:

- OO assumes all requirements are captured during the analysis phase, which is often not the case. This leads to rigid software.
- lack of flexibility
- class libraries tend to grow until costly reorganisation become unavoidable

He reports that companies that pioneered the move to OO now face the evolution of thousands of classes which represents a new kind of legacy system.

As an example, Nokia now sit with complex inheritance hierarchies overloaded with redefinitions. Casais comments that "given the pace at which all economic sectors are taking up OO, an OO reengineering technology is rapidly becoming an acute necessity."

His article presents 18 approaches to reengineering OO systems. (these are refactoring, schema evolution, schema modification primitives, transposed files, filtering and screening, object-oriented views, conversion, class versioning, pattern restructuring, pattern directed reengineering, global reorganisation of hierarchies, hierarchy maintenance, method factorisation,

tailoring and excuses, law of Demeter, visual analysis and metrics-based analysis) However, few of these methods have been tested on real libraries rather than artificial test cases. Nokia is currently busy investigating OO reengineering further. This is bound to become a new important area of research.

7.2.10 Fields where OO will be important

In Pancake's study,[73] several fields of importance were mentioned, indicating market areas where OT is not used yet: design automation, engineering projects, rapid application development, the financial service arena, the manufacturing arena, the Internet, healthcare, and telecommunication.

8. Conclusion

This literature survey aims to highlight the important issues in the transition to object orientation. Technical, human resources as well as corporate issues seem to play a role. The experiences of various authors described here will be used as a guideline in the further research into the success and failure of companies attempting this transition.

9. Reference

[1] Alabiso B. Transformation of data flow analysis model to object oriented design, *OOPSLA '88 Proceedings*, Sept. 1988, pp. 335-353.

[2] Aksit M. and Bergmans L. Obstacles in Object-Oriented Software Development, *OOPSLA '92 Proceedings*, Oct. 1992, pp. 341-358.

[3] Anderson B., Shaw M., Best L. and Beck K. Software Architecture: The Next Step for Object Technology, *OOPSLA '93 Proceedings*, Sept. 1993, pp 356-359. and Panel Session pp. 63-66.

[4] Arnold T.R. and Fuson W.A. Testing "In a Perfect World", *Communications of the ACM*, vol. 37, no. 9, Sept. 1994, pp. 78-86.

[5] Artsy Y.S. Position Paper for OOPSLA Workshop: Are Object-Oriented CASE Frameworks Ready for Prime Time, http://www.compsvcs.com/cs3/artsy.html

[6] Baker M. Position Paper for OOPSLA Workshop: Are Object-Oriented CASE Tools ready for Prime Time, http://www.compsvcs.com/cs3/baker.html

[7] Barclay P.J. and Jackson. S.J. Object-Oriented Programming Transition Strategies, *OOPSLA '93 Proceedings*, Sept. 1993, pp. 39-40.

[8] Barton D. and Arnold T. Evolving to Objects - The Witches' Brew, *OOPSLA '95 Proceedings*, Oct. 1995, pp. 414-425.

[9] Basili V. Briand L.C. and Melo W.L. How reuse influences productivity in object-oriented systems, *Communications of the ACM*, vol. 39, no. 10, Oct. 1996, pp. 104-116.

[10] Beck K. A Laboratory for Teaching Object-Oriented Thinking, *OOPSLA '89 Proceedings*, Oct.1989, pp. 1-6.

[11] Benner K. Position Paper for OOPSLA Workshop: Are Object-oriented CASE Frameworks Ready for Prime Time, http://www.compsvcs.com/cs3/benner.html

[12] Berg W. Cline M. and Girou M. Lessons Learnt from the OS/400 OO Project, *Communications of the ACM*, vol. 38, no. 10, Oct. 1995, pp. 54-64.

[13] Casais E. Re-Engineering Object-Oriented Legacy Systems, *Journal of Object-Oriented Programming*, Jan. 1998, pp. 45-52.

[14] Chambers C. and Ungar D. Making Pure Object-Oriented Languages Practical, *OOPSLA '91 Proceedings*, Oct. 1991, pp. 1-15.

[15] Chidamber S.R. and Kemerer C.F. Towards a metrics suite for object oriented design, *OOPSLA '91 Proceedings*, Oct. 1991, pp. 197-211.

[16] Clyde S.W. Embley D.W. and Woodfield S.N. Tunable formalism in object-oriented systems analysis: meeting the needs of both

theoreticians and practitioners, *OOPSLA '92 Proceedings*, Oct. 1992, pp. 452-465.

[17]Coad P. Finding Objects: Practical Approaches, *OOPSLA '91 Proceedings*, Oct. 1991, pp.17-19.

[18]Cohen J. Manns M. Lilly S. Gabriel R.P. Conway J. and D'Souza D. PANEL: Training Professionals In Object Technology, *OOPSLA '94 Proceedings*, Oct. 1994, pp. 46-50.

[19]Cummins F. Cuniz R. and Lamping J. Next Generation Object-Oriented Programming Languages: Extending the Paradigm, *OOPSLA '93 Proceedings*, Sept. 1993, pp. 91-93.

[20]De Champeaux D. Anderson A. and Feldhousen E. Case Study of Object-Oriented Software Development, *OOPSLA '92 Proceedings*, Oct. 1992, pp. 377-391.

[21]De Champeaux D. and Faure P. A comparative study of object-oriented analysis methods, *Journal of Object Oriented Programming*, Mar/Apr. 1992, pp. 21-32.

[22]De Champeaux D. Balzer B. Bulman D. Culver-Lozo K. Jacobson I. And Mellor S.J. The OO Software Development Process, *OOPSLA '92 Proceedings*, Oct. 1992, pp. 484-489.

[23]De Champeaux D. Baer A.J. Bernsen B. Korncoff A.R. Korson T. and Tkach D.S. Strategies for Object-Oriented Technology Transfer, *OOPSLA '93 Proceedings*, Oct. 1993, pp. 437-447.

[24]De Champeaux D. Constantine L. Jacobson I. Mellor S. Ward P. and Yourdon E. Structured Analysis and Object Oriented Analysis, *ECOOP/OOPSLA '90 Proceedings*, Oct. 1990, pp.135-139.

[25]DeNatale R. The Role of Methods and CASE in OO Development, *OOPSLA '92 Proceedings*, Oct. 1992, pp.39-47.

[26]Dietrich W.C. Nackman L.R. and Gracer F. Saving a Legacy with Objects, *OOPSLA '89 Proceedings*, Oct. 1989, pp. 77-83.

[27]Dodani M. Object-oriented shock therapy, *Journal of Object Oriented Programming*, Jul/Aug. 1996, pp. 17-19.

[28]Fayad M.E. and Tsai W. Object-Oriented Experiences, *Communications of the ACM*, vol. 38, no. 10, Oct. 1995, pp.51-53

[29]Fayad M.E. Tsai W. and Fulghum M.L. Transition to Object-Oriented Software Development, *Communications of the ACM*, vol.39, no.2, Feb. 1996, pp.108-121.

[30]Flensted-Jensen N. Position Paper for OOPSLA Workshop: Are Object-Oriented CASE Frameworks Ready for Prime Time, http://www.compsvcs.com/cs3/neils.html

[31]Fraser S. Beck K. Booch G. Coleman D. Coplien J. Helm R. and Rubin K. How Do Teams Shape Objects – How Do Objects Shape Teams?, 1994, pp. 468-473. and Panel Session pp. 63-66.

[32]Graham I. In Search of the Three Best Books *Journal of Object Oriented Programming*, Sept. 1997, pp. 43-45.

[33]Graham I. Migrating to object technology, Addison-Wesley UK, 1995, pp.3-72. pp.399-444.

[34] Harmon P. The corporate use of object technology, Cutter Consortium, http://www.cutter.com/itgroup/reports/corpuse.html 1997.

[35]Harmon P. Distributed Computing Architecture, Cutter Consortium, http://www.cutter.com/consortium/architecture/corbacom.html, 1998.

[36]Harrison W.H. Shilling J.J. and Sweeney P.F. Good News, Bad News: Experience Building A Software Development Environment Using The Object-Oriented Paradigm, *OOPSLA '89 Proceedings*, Oct. 1989, pp. 85-94.

[37]Hartman M. Jewell F.W. Scott C. and Thornton D. Taking An Object-Oriented Methodology Into The Real World, *OOPSLA '94 Proceedings*, Oct. 1994, pp. 25-30.

[38]Herndon W.R. Sandhu R. and Demurjian S. The Standards Are Coming! Standards For Security In Object-Oriented Systems, *OOPSLA '94 Proceedings*, Oct. 1994, pp. 92-95.

[39]Hill L. Rubin K. Daniels J. Berman C. Coplien J. and Johnson D. Managing Object Oriented Projects, *OOPSLA '95 Proceedings*, Oct. 1995, pp.88-90.

[40]Høydalsvik G.M. and Sindre G. On The Purpose Of Object-Oriented Analysis, *OOPSLA '93 Proceedings*, Oct. 1993, pp. 240-255.

[41]Jacobson I. and Lindstrom F. Re-engineering of old systems to an object-oriented architecture, *OOPSLA '91 Proceedings*, Oct. 1991, pp. 340-350.

[42]Kerth N.L. A Structured Approach To Object-Oriented Design, *OOPSLA '91 Proceedings*, Oct. 1991, pp.21-43.

[43]Korson T. Managing the Transition to Object-Oriented Technology, *OOPSLA '91 Proceedings*, Oct. 1991, pp. 55-62.

[44]Korson T. The Role Of A Corporate Object Technology Center, *OOPSLA '93 Proceedings*, Oct. 1993, pp.131-134.

[45]Kristek T. and Vaishnavi V. Role of a Corporate Technology Center, *OOPSLA '94 Proceedings*, Oct. 1994, pp. 72-77.

[46]Laffra C. and Malhotra A. Advanced Techniques For Understanding Profiling And Debugging Object Oriented Systems, *OOPSLA '93 Proceedings*, Oct. 1993, pp. 79-81.

[47]Lato K. Learn to learn: Training on new technology, *Journal of Object Oriented Programming*, Mar/Apr. 1997, pp. 24-27.

[48]Laurance D.C. Position Paper for OOPSLA Workshop: Are Object-Oriented CASE Frameworks Ready For Prime Time, http://www.compsvcs.com/cs3/att.html

[49]Lee W. How To Adapt OO Development Methods In A Software Development Organisation – A Case Study, *OOPSLA '94 Proceedings*, Oct. 1994, pp. 19-24.

[50]Li W. Henry S. Kafura D. and Schulman R. Measuring object-oriented design, *Journal of Object-Oriented Programming*, Jul/Aug. 1995, pp. 48-55.

[51]Litvintchouk S.D. Evolving Towards Object-Oriented Technology In Large Organizations, *OOPSLA '93 Proceedings*, Oct. 1993, pp. 73-76.

[52]Liu C. Goetze S. and Glynn B. What Contributes To Successful Object-Oriented Learning, *OOPSLA '92 Proceedings*, Oct. 1992, pp. 77-86.

[53]Love T. Avoiding Uninteresting Old Mistakes, *Journal of Object Oriented Programming*, Jul/Aug 1997, pp. 14-15.

[54]Malan R. Coleman D. and Letsinger R. Lessons from the Experiences of Leading Edge Object Technology Projects in Hewlett-Packard, *OOPSLA '95 Proceedings*, Oct. 1995, pp. 33-46.

[55]Malloy M.A. Post-Mortem Assessment of Interface Changes for an Evolving Object-Oriented, "Not-So-Rapid" Prototype, *OOPSLA '93 Proceedings*, Oct. 1993, pp. 17-18.

[56]Manns M.L. and Nelson H.J. Retraining procedure-oriented developers: An issue of skill transfer, *Journal of Object Oriented Programming*, Nov/Dec. 1996, pp. 6-10.

[57]Martin R. and Ottinger T. Limitations of OO? Usenet Discussion Commentary, *Object Magazine Online*, http://www.objectmagazine.com/tips/9711/martin.html

[58]Martin R. and Ottinger T. Limitations of OO? Usenet Discussion Commentary Part 2, *Object Magazine Online*, http://www.sigs.com/omo/tips/martin.html

[59]Meszaros G. Experience Building Large OO Frameworks at BNR, *OOPSLA '94 Proceedings*, Oct. 1994, pp. 14-18.

[60]Meyer B. Towards an O-O Curriculum, http://www.tools.com/doc/manuals/technology/curriculum/index.html.

[61]Meyers G.L. Position Paper for OOPSLA Workshop: Are Object-Oriented CASE Frameworks Ready for Prime Time, http://www.compsvcs.com/cs3/meyers.html

[62]Moreau D.R. and Dominick W.D. Object-Oriented Graphical Information Systems: Research Plan and Evaluation Metrics, *The Journal of Systems and Software*, vol.10, 1989, pp.23-28.

[63]Narayanaswamy K. Are Object Oriented CASE Frameworks Ready for Prime Time, *OOPSLA '95 Proceedings*, Oct. 1995, pp. 155-158.

[64]Noble J. and Kolbe K Position Paper: Are Object Oriented CASE Frameworks Ready for Prime Time, http://www.compsvcs.com/cs3/noble.html

[65]Object Magazine Online, The Object Magazine Online Interview - James Coplien, 1998, http://www.objectmagazine.com

[66]Object Magazine Online, The Object Magazine Online Interview -Bjarne Stroustrup, 1998, http://www.objectmagazine.com

[67]The Object Magazine Online Survey: What are you using?, Jan. 1998, http://www.sigs.com/omo/questionnaire/results.html

[68]The Object Magazine Online Survey: What's Your Biggest Concern?, January 1998, http://www.objectmagazine.com/questionnaire/9801/results.html

[69]Olander G. Experience Report - Chembench: Redesign of a Large Commercial Application Using Object-Oriented Techniques, *OOPSLA '92 Proceedings*, Oct. 1992, pp. 13-16.

[70]Ovum Evaluates: CASE Products, Why size is important, Oct. 1995, http://www.ovum.com/evaluate/case/index.html

[71]Page-Jones M. Object Orientation: Making the Transition, copyright Wayland Systems Inc. 1997, pp. 1-19.

[72]Page-Jones M. The seven stages in software engineering, *American Programmer*, Jul/Aug. 1990.

[73]Pancake C.M. The Promise and the Cost of Object Technology: A five-year forecast, *Communications of the ACM*, Oct. 1995, vol.38, no.10, pp.33-49.

[74]Pickering C. Survey of advanced technology, copyright Systems Development Inc., 1993.

[75]Pircher P. Position Paper for OOPSLA Workshop: Are Object-Oriented CASE Frameworks Ready for Prime Time, http://www.compsvcs.com/cs3/pircher.html

[76]Reed D.R. Cagan M. Goldstein T. and Moo B. Issues in Moving from C to C++, *OOPSLA '91 Proceedings*, Oct. 1991, pp. 163-165.

[77]Riehle R. Reuse OOP and Safety-Critical Software, *Journal of Object Oriented Programming*, Nov/Dec. 1997, pp. 21-24.

[78]Rivas E.A. Position Paper for OOPSLA Workshop: Are Object-Oriented CASE Frameworks Ready for Prime Time, http://www.compsvcs.com/cs3/rivas.html

[79]Rosson M. and Carroll J.M. Scaffolded examples for learning object-oriented design, *Communications of the ACM*, vol. 39, no. 4, Apr. 1996, pp. 46-47.

[80]Schach SR. Classical and Object-oriented software engineering, third edition, Irwin/McGraw-Hill, USA 1996, pp. 70-74.

[81]Schedlbauer M. How to Select a Training Organisation, *Journal of Object Oriented Programming*, Oct. 1997, pp. 39-40.

[82]Seldon A. The basics of object standards, *Computerweek*, Aug. 1997 p. 16.

[83]Shan Y-P. Morgan T. Proudfoot P. Thompson J. Tibbetts J. and Woolfrey A. Objects on the

Server: Are We Ready, *OOPSLA '96 Proceedings*, Oct. 1996, pp. 384-388.

[84]Shan Y-P. Auer K. Bear A.J. Adamczyk J. Goldberg A. Love T. and Thomas D. Smalltalk in the Business World the Good the Bad and the Future, *OOPSLA '94 Proceedings*, Oct. 1994, pp. 145-152.

[85]Shih T.K. Wang C-C. and Chung C-M. Using Z to specify object-oriented software complexity measures, *Information and Software Technology*, vol.39, 1997, pp. 515-529.

[86]Stark M. Impacts of object-oriented technologies: seven years of SEL studies, *OOPSLA '93 Proceedings*, Oct. 1993, pp. 365-373.

[87]Townsend P. and Murphy G. Experience Report -- Objects in the Life-Cycle, *OOPSLA '92 Proceedings*, Oct. 1992, pp. 25-28.

[88]Vaishnavi V. and Korson T. Role of a Corporate Object Technology Center, *OOPSLA '95 Proceedings*, Oct. 1995, pp.186-190.

[89]Vayda T.P. Lessons From the Battlefield, *OOPSLA '95 Proceedings*, Oct. 1995, pp. 439-452.

[90]Viljoen P. Object Orientation: Concepts and Method, version 4, copyright Infomet, 1992, pp.1-123.

[91]Wick M.R. On using C++ and Object-Orientation in CS1: The message is still more important than the medium, *SIGCSE '95*, Mar. 1995, pp. 322-326.

[92]Yourdon E. Object-oriented analysis and design seminar, 1994, pp. 1-150.

10. Author Contact Details

Contact details: **Miranda Jansen van Rensburg**, Technikon SA (Information Technology), C de Wet Drive Roodepoort Office Phone: +27-11-471-2929, Fax: +27-11-471-3270, Internet E-mail: mjvanren@tsamail.trsa.ac.za

■



NDM QMS

NDM - The Research Process

Technical Product

Version 1.00

Document Status: Approved

Table of Contents

Table of Contents ii

Change History iii

Configuration Control iii

Document History iii

Revision History iii

Management Authorisation iii

Change Forecast iii

1 Scope 1

1.1 Introduction 1

1.2 Audience 1

1.3 Applicable Documents 1

1.4 Requirements Traceability 1

1.5 Abbreviations 1

2 The process 2

Change History

Configuration Control

Project:	NDM QMS
Title:	NDM - The research process
Doc. Reference:	C:\MSC\TP\NDM309.100
Created by:	M Jansen Van Rensburg
Creation Date:	10 July 1998

Document History

Version	Date	Status	Who	Saved as:
0.01	98\07\10	Draft	MVR	NDM309.010
1.00	98\09\23	Approved	MVR	NDM309.100

Revision History

Version	Date	Changes
0.01	98\07\10	Created using NDM002.010
1.00	98\09\23	Approved

Management Authorisation

Version	Date	Status	Project Minute Reference
1.00	98\09\23	Approved	98\09\23 3.

Change Forecast

1 Scope

1.1 Introduction

This document describes the research process that was followed for the research project Pitfalls and Guidelines in the transition to object oriented software design methodologies.

1.2 Audience

The audience for this document comprise the various stakeholders of the SEAL, i.e.:

- Head of the Department, Electrical Engineering
- Product developer M Jansen van Rensburg
- Product Manager and supervisor Prof Dwolatzky

1.3 Applicable Documents

1.3.1 SEAL QMS Standards

SEAL QMS Document Layout, Presentation and Typesetting Guide, QS 003, Revision 1.00, 3 October 1994

1.3.2 International Standards

a. ISO\IEC 15504 Process Assessment Part 2: A Reference Model For Processes And Process Capability Date: PDTR, October 1996

b. ISO\IEC 9126 (1991) -Information Technology - Software product evaluation - Quality characteristics and guidelines for their use.

1.4 Requirements Traceability

a. ISO 9001 (1994) Clause 4.2

1.5 Abbreviations

OO Object Orientation

2 The process

The research was structured as follows:

Firstly, a literature survey was done to find out what is already known about this research area. At the same time a few informal interviews were conducted with selected South African companies to determine what the issues involved in a transition to Object Orientation are.

Secondly, once the issues were known, a telephonic interview was held with 120 South African companies in order to find a selection of companies that could be used for the final interview. The questions used for this interview were formulated as Yes No or multiple choice questions, so that statistics regarding the progress towards OO could be obtained. This telephonic interview also served to highlight certain trends in the OO arena in South Africa.

Further interviews were held with 12 South African companies chosen from the original 120 companies. This interview delved more deeply into the issues that have arisen from the first informal interview, the literature study as well as the telephonic interviews. It served to describe the typical situation in South African companies and was also used to verify certain trends that were found during the telephonic interviews. Finally it extracted the opinions of the respondents regarding future trends for OO. In contrast to the telephonic interview, questions in this interview were posed as issues for discussion rather than simple Yes No questions.

Lastly, once it became clear what the situation regarding OO in South Africa was, these results were compared with a selection of companies in the USA, to create a clear view of the differences and similarities locally and abroad.

The conclusions reached can be used to guide South African companies in their transition to OO.

■



NDM QMS

NDM - The Informal Interviews: Conclusions

Technical Product

Version 1.00

Document Status: Approved

Table of Contents

Table of Contents ii

Change History iii

Configuration Control iii

Document History iii

Revision History iii

Management Authorisation iii

Change Forecast iii

1 Scope 1

1.1 Introduction 1

1.2 Audience 1

1.3 Applicable Documents 1

1.4 Requirements Traceability 1

1.5 Abbreviations 1

2 Summary of the first, informal interviews 2

2.1 The companies 2

2.2 The interviews 2

Change History

Configuration Control

Project:	NDM QMS
Title:	NDM - The informal interviews: conclusions
Doc. Reference:	C:\MSC\TP\NDM305.100
Created by:	M Jansen Van Rensburg
Creation Date:	2 April 1998

Document History

Version	Date	Status	Who	Saved as:
0.01	98\04\02	Draft	MVR	NDM305.010
1.00	98\04\22	Approved	MVR	NDM305.100

Revision History

Version	Date	Changes
0.01	98\04\02	Created using NDM002.010
1.00	98\04\22	Approved

Management Authorisation

Version	Date	Status	Project Minute Reference
1.00	98\04\22	Approved	98\04\22 2.4

Change Forecast

1 Scope

1.1 Introduction

This document provides some conclusions that were reached after interviews were held with four South African companies during March and April 1998. These interviews were held at the start of the research project that investigates the pitfalls and guidelines in the transition to object oriented software design methodologies.

1.2 Audience

The audience for this document comprise the various stakeholders of the SEAL, including:

- Head of the Department, Electrical Engineering
- Product developer M Jansen van Rensburg
- Product Manager and supervisor Prof Dwolatzky

1.3 Applicable Documents

1.3.1 SEAL QMS Standards

SEAL QMS Document Layout, Presentation and Typesetting Guide, QS 003, Revision 1.00, 3 October 1994

1.3.2 International Standards

a. ISO\IEC 15504 Process Assessment Part 2: A Reference Model For Processes And Process Capability Date: PDTR, October 1996

b. ISO\IEC 9126 (1991) -Information Technology - Software product evaluation - Quality characteristics and guidelines for their use.

1.4 Requirements Traceability

a. ISO 9001 (1994) Clause 4.2

1.5 Abbreviations

OO Object Orientation

2 Summary of the first, informal interviews

The interviews were held at the start of the research, to find out what the issues are involved in making the transition to Object Orientation. Interviews were held in person with representatives from four companies, that had experience in the transition to OO and would be able to help in highlighting the issues that would be investigated more thoroughly in the rest of the research. The average duration of each interview was approximately one hour. Questions were formulated as issues for discussion rather than simple yes no or multiple choice questions.

2.1 The companies

2.1.1 Spescom

The interview was held on 11 March 1998 with Viv Crone, group technical director, who became involved in OO in 1987 when OO was still very new in South Africa.

2.1.2 BSW-Data Ltd.

The interview was held on 17 March 1998 with Andre Baas, a director at BSW-Data Ltd. He was involved in a network management project at BSW, which was done with a view on reusability using OO.

2.1.3 SPL

In this case the interview was held on 27 March 1998 with Malcolm Rabson who is responsible for the Object Technology initiative at SPL.

2.1.4 ABSA

The interview was held on 20 April 1998 with Dr Conor Hughes, a technology consultant at ABSA, who was brought in to introduce OO into the company. At the time ABSA had claimed to already have adopted OO but, in fact, IBM did the core work on the major project which was using OO. ABSA only implemented the interface to the legacy systems.

2.2 The interviews

The issues that these four companies mentioned were consolidated and served to highlight the questions that needed to be asked in the future

questionnaires. Where applicable, the company where the comment originated is given in brackets.

2.2.1 A different situation from overseas

The situation here in South Africa is very different from overseas. The following comments lead to this conclusion:

- In 1987, due to the sanctions, it was very difficult to get information about OO. OO was also too new for the time. (Spescom)
- Due to the "skills drain" there is a loss of expertise in South Africa as well as a skills shortage. People are pulled into the USA market because South African employees are generally trained on a wider variety of topics and are hardworking.
- Retaining skills has become an issue: salaries are pushed up by the skill shortage, due to emigration to the USA rather than the competition experienced locally (BSW)
- because of skill shortage, training shortage etc., South Africa will only support major technologies; it is too expensive to diverge. (BSW)
- In South Africa crisis management often takes place just to "get things done"
- Due to the nature of the business, clients demand projects faster nowadays, which means that OO only gets implemented if the requirement is "state of the art".
- software is being imported which makes the need for new local development less. (BSW)
- t the bad exchange rates it is difficult to buy CASE tools from overseas (BSW)
- There seems to be a lack of understanding: people often claim to write objects but are actually doing functional decomposition. (BSW)
- systems can be built cheaper overseas (SPL)
- there is no more limitless money available for OO research.(BSW)
- here are more companies in the USA so even if 10 companies made a success of OO it still means that only for instance 0.01% made a success, which implicates that perhaps South Africa is not as far behind the rest of the world as people think. (SPL)

2.2.2 Training

- the education in South Africa is to blame - in many institutions there is no teaching of OO (SPL)
- The percentage time spent on various phases of a project needs to be understood which is why the project manager also needs to attend the training. (ABSA)
- there is no cookbook available to implement OO which makes it difficult.(Spescom)

2.2.3 The time when moving to OO

From the following comments it seemed as though the time when moving to OO plays a role in the success or failure of the process:

- There is a concern about OO in real time development – although perhaps today it is not a problem any more with today's faster processors (Spescom)
- In BSW's case there were limited CASE tools available at the time their OO project was developed.
- From 1992 until 1994 there was a lot of movement towards OO but perhaps it has now reached a plateau (BSW)
- regarding project management, there have been no cost and time estimating tools for OO, whereas there always were for the structured methodologies (BSW)

2.2.4 Market sector

- In the military environment there were methodologies in place even though these were structured methodologies – this seems to be lacking with OO. (Spescom)
- According to Spescom, the IT business was previously divided into 3 sectors: military (based on performance levels, well-motivated people), industrial (less formal, faster methods), and banking. The military sector seems to have shrunk now.
- the domain can also influence the decision - in telecommunication there have not been a lot of OO libraries available.(BSW)

2.2.5 Management

On the question of management, there seems to be some doubts about management's knowledge of OO (especially where people had years of Cobol experience). For this reason it seemed as though the issue of management would have to be explored further in the following interviews.

- there is no support for OO from management (in the case of the sortware house) which implies that the situation must then be worse in the case of other types of companies. (SPL)
- At ABSA a case was mentioned where the manager forced the use of OO which seemed to work successfully.

2.2.6 Reuse

At BSW the project undertaken had as its goal achieving reuse, however, there seemed to be disappointment that not as much reuse was achieved on the first project as originally thought, although the culture for future reuse had been established.

- accessibility is a problem - if there is no way to browse the objects already developed, reuse will not happen.
- people like to build systems themselves, therefore the reward structure should change for reuse to be successful
- ABSA created macro objects for reuse

2.2.7 The first project

- OO requires a lot of training – therefore the recommendation is do not implement it fully on the first project. (BSW)
- the pilot project should be small but should have time scales attached to make sure things get done. (BSW)
- the first change is often quite big; developers have to throw away all code and then start again - this can be demotivating (BSW)
- 3 projects and 1 pilot project was developed using OO (SPL)
- In ABSA, a small non-critical project was developed which was successful.

2.2.8 Company size

- OO requires a huge amount of investment, therefore company size also influences the progress. (BSW)
- In small companies it is easier to make the transition due to less bureaucracy. A bureaucratic culture stops the company from using small teams (ABSA)

2.2.9 Quality

- "The professionalism of Software Engineering in South Africa has not progressed" (Spescom)
- regarding ISO9000 there was no priority at the time to get ISO9000 in the case of BSW – the opinion is that it must be difficult to achieve that and move to OO at the same time. One should get ISO 9000 accreditation first and then move to OO.
- As with BSW, there was no pressure to get ISO 9000 accreditation in SPL; it only applicable when dealing with overseas companies.
- the opinion is that OO and ISO 9000 are mutually exclusive(SPL)
- while the project was developed, there were lots of iterations in the object modelling, a few objects were implemented and then the whole design would change again. This indicates that at least an iterative development process needs to be in place (BSW)
- there is no relationship between ISO 9000 and the successful adoption of OO. (ABSA)

2.2.10 CASE tools

- Andre Baas from BSW recommends rather omitting CASE tools and getting key people.
- ABSA used Objectory, which became part of Rational Rose, and also used use cases. They were also investigating Cool tools e.g. Cooljex and are still busy evaluating it.

2.2.11 Databases

Regarding databases, the market still demands Oracle even though it is not Object Oriented.

2.2.12 Guidelines for adopting OO

- Focus not on how clever you can get with OO, no one is prepared to wait for solutions, projects are getting shorter (3-6 months) and the competition is getting worse (BSW)
- The important factors involved are attitude, having an OO guru and having a project manager that also attended the training in order to understand the timescales involved.(ABSA)
- An experienced solutions architect is essential for success (ABSA)
- Domain knowledge is important: in the case of BSW there was already a previous project developed in this technology so that the technology itself was not also new (BSW)

2.2.13 Languages

- C++ added lots of problems with memory leaks, it was difficult to find a memory management tool at the time (BSW)
- Java is not in their domain so it is not used (BSW)
- BSW rather use what the market dictates which was and still is C++.
- there is a lot of C++ development taking place but whether it is truly OO based is uncertain (BSW)
- at SPL they decided on Smalltalk for the following reasons: C++ does not impose OO and programmers were making mistakes regarding memory management whereas Smalltalk offers garbage collection.
- Java will get world-wide adoption because of its platform independence, Visual Basic runs on only one platform and offers no security (SPL)
- Languages used in ABSA are: Java, Smalltalk , and C++. Smalltalk will however fall away since distributed Smalltalk has never really taken off.

2.2.14 The current state of OO

The following comments were made:

- OO is still considered to be a research project (SPL)

- OO knowledge and experience is often strategic and companies do not want to share their knowledge(SPL) The comment "People doing little talk a lot, people who do the work stay quiet" describes this issue.

- According to SPL, a third of companies will make the transition to OO, a third will stay with COBOL and a third will be put out to pasture!

2.2.15 Influence of the Year 2000 Problem

In the case of ABSA, as soon as people become free from projects they are put on the Y2K problem, which causes a delay in the move to OO.

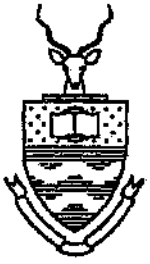
2.2.16 Resistance

Regarding attitude, some people do not want to change; they feel safe with what they know and what they know is enough to ensure work for lots of years to come (ABSA)

There should be a willingness to get it wrong and try again. (ABSA)

No technical problems were experiences since the people were all young and wanted to learn (ABSA)

■



NDM QMS

NDM - List of Companies

Technical Product

Version 1.00

Document Status: Approved

Table of Contents

Table of Contents ii

Change History iii

Configuration Control iii

Document History iii

Revision History iii

Management Authorisation iii

Change Forecast iii

1 Scope 1

1.1 Introduction 1

1.2 Audience 1

1.3 Applicable Documents 1

1.4 Requirements Traceability 1

2 List of companies 2

Change History

Configuration Control

Project:	NDM QMS
Title:	NDM - List of companies
Doc. Reference:	C:\MSC\TP\NDM301.100
Created by:	M Jansen Van Rensburg
Creation Date:	24 April 1998

Document History

Version	Date	Status	Who	Saved as:
0.01	98\04\24	Draft	MVR	NDM301.010
1.00	98\05\06	Approved	MVR	NDM301.100

Revision History

Version	Date	Changes
0.01	98\04\24	Created using NDM002.010
1.00	98\05\06	Approved

Management Authorisation

Version	Date	Status	Project Minute Reference
1.00	98\05\06	Approved	98\05\06 3.

Change Forecast

1 Scope

1.1 Introduction

This document provides a record of the list of companies used for the telephonic interviews conducted during May 1998. These interviews were conducted as part of the research report investigating the pitfalls and guidelines in the transition to object oriented software design methodologies.

1.2 Audience

The audience for this document comprise the various stakeholders of the SEAL, including:

- Head of the Department, Electrical Engineering
- Product developer M Jansen van Rensburg
- Product Manager and supervisor Prof Dwolatzky

1.3 Applicable Documents

1.3.1 SEAL QMS Standards

SEAL QMS Document Layout, Presentation and Typesetting Guide, QS 003, Revision 1.00, 3 October 1994

1.3.2 International Standards

a. ISO\IEC 15504 Process Assessment Part 2: A Reference Model For Processes And Process Capability Date: PDTR, October 1996

b. ISO\IEC 9126 (1991) -Information Technology - Software product evaluation - Quality characteristics and guidelines for their use.

1.4 Requirements Traceability

a. ISO 9001 (1994) Clause 4.2

2 List of companies

The market sectors that were used include the following areas:

Description	Market sector
Financial / banking	Financial
Travel / tourism / hotels / restaurant	Hospitality
Printing / publishing / media	Media
Mining	Mining
Manufacturing / engineering / construction / process, etc.	Manufacturing
Service / panel beating / electrical contractor / security etc.	Service
Government	Government
Health	Health
Retail	Retail
Education	Education
Software / Information Technology / Telecommunication / Electrical	Information Technology (IT)
Other	Other

The following list of companies were obtained using various Internet and other business directories.

Where not specified, the dialling code is (011).

Market Sector	Company	Telephone Number	Contact Person
Financial			
	Prestasi	6526725	Bernard Niewoudt
	Fedsure	3226000	Dave Butler
	Southern Life	021-6580911	Rob Kruger
	Momentum	012-6718911	Carl Conradie
	Future Bank	4818300	Amanda Muer
	Unibank	8067600	Dirk Otto
	Saambou	012-4215000	Elise Swanepoel
	Legal Wise	4704000	Mr Prior
	BOE	3021000	Jerry Comninos
	Huysamer Stals	2403500	Doug Mar
	Reserve Bank	012-313-3911	Rob De Jong
	UAL Merchant Bank	4801000	Fazel Mayett
	Wesbank	8368181	Craig Myburg
	RMB	2828000	Vincent Coetzee
	Stanbic	3586700	Chris Blake
	NBS	031-3641111	Henk Lategan
	Alexander Forbes	2690000	Len Bergen
	Liberty Life	4083911	Beatrice Bayes
	Santam	021-9157000	Trevor Earnston
	New Republic Bank	031-3047544	Ahmed Muslim

Market Sector	Company	Telephone Number	Contact Person
Education			
	Radical Training	031-844933	
	ECP		Anette
	Birnam	8872540	Patrick Masdon
	TSA	4712000	
	Redhill Prep School	7834707	E v Wouw
	Department Of Education	012-3125911	Hudson
Government			
	Spoornet	7744666	Rian Oosthuisen
	SABS	012-4287911	Jakkie Pretorius
	Municipality Randburg	7890556	Brian Germishuys
	SPCA Randburg	4621610	Zina
Health			
	Adcock	9211511	Bi Jugmahan
	Wilgers	012-8070019	Ms Strydom
	Medscheme	7879607	David Cloos
	Kirsch Pharma	3925171	Tersia
	Glaxo	3136000	Trevor Matthews
	Brackenfell Hospital	021-9814547	L Rutter
Hospitality			

Market Sector	Company	Telephone Number	Contact Person
	Satour	012-3470600	Gerda Coetzee
	O'Hagans	7871108	Colleen V Tonder
	Southern Sun	7800200	Amos Kahn(Vircom)
	Seekers	3212400	Gary Web
	Imperial	4530005	Richard Henwood
	SA Breweries	4071700	Kerry Strydom
	Coaches For Africa	031-5615663	Charmaine
Information Technology			
	Internet Solution	2835000	Nithal Goburdham
	Computer Foundation	012-672-0100	Eugene Marais
	SPL	3222165	Malcolm Rabson
	QData	2666232	Johan De Beer
	Software Futures	8071340	Ahmed Chicktay
	Cenit	012-672-0000	Johan Smook
	Usko	8070777	Julian
	Altech Smart Card	8043226	Leo Murray
	Vircom	8072333	Amos Kahn
	Siemens	4889111	George Engels
	Datatec	2333344	Bruce Tailor
	Abstraction	8821918	Gary

Market Sector	Company	Telephone Number	Contact Person
	ICL	8024281	
	Spescom	2861500	Viv Crone
	Brainware	012-6635677	Allen Mcneal
	Paradigm	012-6725700	Peter Koolman
	AKS	8862225	Brian Loudon
	Energy Measurements	9217900	Keith Richards
	Objectsoft	7878631	Thys Brits
	Elkon	012-6635677	Trevor Van Rensburg
Manufacturing			
	BP	021 4082911	John Coxwell
	Eskom	8005823	Mike Rod
	Temsa	8132220	Tom V Zyl
	Cargo	6243700	Phyllis
	Tiger Oats	8844500	Croydon Coppings
	Outspan	012-6635100	Hennie Boshoff
	Plasserail	4741541	
	VKE	012-4813800	Susanna Jordaan
	Nampak	083-273-1419	Miles James
	Wastetek	4565400	Rita V Wyk
	ESD	6525555	Margot De regt
	AEC	3164911	Dr Van der Walt

Market Sector	Company	Telephone Number	Contact Person
	BMW	3383000	Bennie Vorster
	BIFSA	8051985	Pierre Fourie
	Murray Roberts	4553101	Daan Vermaas
	Kentron	012-671-1155	Frans Roodt
	Mondi:Recycling	6231583	Ronnie Dittirich
	Dorbyl Marine	031-251511	Craig Samuel
	Nanotek	012-665-1338	Addie Buissinne
	Polifin	3296111	Annemarie Diedricks
	Amalgamated Beverages	4554020	Francois
	Prieska Engineering	59461382	Mrs Marx
	Burger And Wallace	021-9053660	Hanlie
Media			
	Multichoice	2893000	Neville Gubb
	Getaway	021-5311391	Maureen Van Niekerk
	Computicket	4458100	Rob Mills
	Highveld ridge Newspaper	017-6347728	DJ
	Primedia Broadcasting	8848400	Andrew Brooks
Mining			

Market Sector	Company	Telephone Number	Contact Person
	Mintek	7094111	Andy Shipman
	RBCT	0351-904911	Chris Meyer
	RBM	0351-9013131	Dave Cox
	Billiton	3769111	IT Dept
	Bastion(ISCOR)	012-3073000	Tertius Cloete
	Samat Mining	9742051	Mike Watts Farmer
Retail			
	Koljander	7266282	Elaine
	Game	031-302-8991	Dyan
	Checkers	021-980-4000	Tom Roos
	Juicy Lucy	8037500	Peter Hollis
	Edgars	4956000	Andre Oberholzer
	Foto First	8871600	Deon Serfontein
	EMI	4064000	Koble Pearson
	C N A	4917612	Gracia Valenti
	PicknPay	021-6581612	Abe Malana
	Angela's Hair Boutique	6484293	Angela
Service			
	Anderson	3283000	Kirsten Hardy
	IMM	4821419	Tobi
	Marketel	4021640	Henri Rex

Market Sector	Company	Telephone Number	Contact Person
	AOS	836278920	Neville Ross
	Safmarine	021-408-6290	Alida Riddell
	Twines	7931161	Mccarthy
	Coin Security	012-8001211	Gawie Jv Rensburg
	Optimark	021-9485212	Chantal
	CPL		Chris Hays
	Teljoy	8526000	Stein De Villiers
	Online Personnel	041-352266	Nicola



NDM QMS

NDM - Telephonic Questionnaire

Technical Product

Version 1.00

Document Status: Approved

Table of Contents

Table of Contents ii

Change History iii

Configuration Control iii

Document History iii

Revision History iii

Management Authorisation iii

Change Forecast iii

1 Scope 1

1.1 Introduction 1

1.2 Audience 1

1.3 Applicable Documents 1

1.4 Requirements Traceability 1

2 Telephonic Questionnaire 2

Change History

Configuration Control

Project:	NDM QMS
Title:	NDM - Telephonic Questionnaire
Doc. Reference:	C:\MSC\TP\NDM302.100
Created by:	M Jansen Van Rensburg
Creation Date:	5 May 1998

Document History

Version	Date	Status	Who	Saved as:
0.01	98\05\06	Draft	MVR	NDM302.010
1.00	98\05\06	Approved	MVR	NDM302.100

Revision History

Version	Date	Changes
0.01	98\05\06	Created using NDM002.010
1.00	98\05\06	Approved

Management Authorisation

Version	Date	Status	Project Minute Reference
1.00	98\05\06	Approved	98\05\06 3.

Change Forecast

1 Scope

1.1 Introduction

This document provides a record of the questionnaire used during telephonic interviews conducted during May 1998.

1.2 Audience

The audience for this document comprise the various stakeholders of the SEAL, including:

- Head of the Department, Electrical Engineering
- Product developer M Jansen van Rensburg
- Product Manager and supervisor Prof Dwolatzky

1.3 Applicable Documents

1.3.1 SEAL QMS Standards

SEAL QMS Document Layout, Presentation and Typesetting Guide, QS 003, Revision 1.00, 3 October 1994

1.3.2 International Standards

a. ISO\IEC 15504 Process Assessment Part 2: A Reference Model For Processes And Process Capability Date: PDTR, October 1996

b. ISO\IEC 9126 (1991) -Information Technology - Software product evaluation - Quality characteristics and guidelines for their use.

1.4 Requirements Traceability

a. ISO 9001 (1994) Clause 4.2

2 Telephonic Questionnaire

Company: _____

Telephone Number: _____

Name: _____

Is it the head office? Get head office number.

Ask to speak to the IT manager?

If there is no IT manager then speak to the General Manager?

Get his / her name.

My name is Miranda van Rensburg, I am busy with a MSc at Wits University. I would like to ask you some questions regarding software development, which will take about 5 minutes. The information will be held in confidence. Can you talk now or should I perhaps phone back at a later stage?

Please select only one option in each case that most applies to your company.

1. Where is your company based?

- ☐ National (with regional offices in most parts of SA)
- ☐ Gauteng
- ☐ Western Cape
- ☐ Kwazulu Natal Eastern Cape
- ☐ Other

2. How would you describe the market sector that your company falls in?

_____ (example financial, mining, etc.)

3. What is the total company size?

- ☐ 10 people or less
- ☐ 11 – 100 people
- ☐ 100 – 1000 people
- ☐ more than a 1000

4. Does software play any role in your company? (do you use or develop software)

- ☐ Yes
- ☐ No

IF THE ANSWER IS NO THEN DO NOT GO ON.

"Thank you for your time. Bye"

5. Do you have an IT department that supports other departments or is IT the company's major business?

- ☐ Supports other departments
- ☐ IT is the major business

6. What is the size of the IT department (number of people)

- ☐ 1 person or none
- ☐ 2 – 10 people
- ☐ 11 – 20 people
- ☐ More than 20 people (specify _____)

7. What is the skill level of the IT group in your company?

- ☐ Mostly university degrees
- ☐ More than half have university or technikon degrees
- ☐ Mostly attended courses such as CNE, MCSE (specify _____)
- ☐ Mostly self trained (or on the job)
- ☐ None of the above / not applicable

8. What is your company's level of exposure to Object Oriented Technology?

- ☐ Have heard of Object Orientation
- ☐ Have used an object oriented product
- ☐ Have implemented Object Orientation – number of years:

  _____
- ☐ Have used advanced techniques such as CORBA and patterns
- ☐ None of the above

9. Does your company have ISO 9000 accreditation or similar, or currently have a Quality plan or procedure for software?

- ☐ Yes, all departments have
- ☐ Most departments have
- ☐ Less than half of the company has
- ☐ In the process of getting accreditation
- ☐ No accreditation or plan

10. What is the average software project size?

- ☐ 1 person
- ☐ 2 – 5 people
- ☐ 5 – 10 people
- ☐ more than 10 people
- ☐ it varies (please specify _____)

11. What is the average lifetime of software projects in your company?

- ☐ Less than 3 months
- ☐ 3 – 6 months
- ☐ 6 months – 1 year
- ☐ more than 1 year
- ☐ it varies (please specify _____)

12. What is the size of your company's IT budget? Categories are:

- ☐ Would prefer not to answer
- ☐ Less than R 100 000
- ☐ R 100 000 – R 1million
- ☐ R 1 million – R 10 million
- ☐ R 10 million – R 50 million
- ☐ More than R 50 million

13. How would you describe the nature of your company's IT business?

- ☐ Research and Software development
- ☐ Sales / Marketing
- ☐ Consulting
- ☐ Support
- ☐ None of the above, specify _____

14. Do you use any methodologies or CASE tools for software development?

- ☐ Always
- ☐ Almost always
- ☐ Almost never
- ☐ Never

Please specify which tools you use: _____

15. Would your company be prepared to participate in a further interview to discuss software development?

- ☐ Yes
- ☐ No

16. Are you the right person to speak to about Object Technology or is there perhaps someone else?

- ☐ Yes
- ☐ No, speak to _____

17. Would you like to see a copy of the report that will be generated about the research?

- ☐ Yes
- ☐ No

Postal Address:

_____

_____

"Thank you for your time. Bye"



NDM QMS

NDM QMS - Results From Telephonic Interviews

Technical Product

Version 1.00

Document Status: Approved

Table of Contents

Table of Contents ii

Change History iii

Configuration Control iii

Document History iii

Revision History iii

Management Authorisation iii

Change Forecast iii

1 Scope i

1.1 Introduction i

1.2 Audience i

1.3 Applicable Documents i

1.4 Requirements Traceability i

2 The results i

Change History

Configuration Control

Project:	NDM QMS
Title:	NDM QMS - Results from telephonic interviews
Doc. Reference:	C:\MSC\TP\NDM306.100
Created by:	M Jansen Van Rensburg
Creation Date:	10 May 1998

Document History

Version	Date	Status	Who	Saved as:
0.01	98\05\10	Draft	MVR	NDM306.010
1.00	98\06\01	Approved	MVR	NDM306.100

Revision History

Version	Date	Changes
0.01	98\05\10	Created using NDM002.010
1.00	98\06\01	Approved

Management Authorisation

Version	Date	Status	Project Minute Reference
1.00	98\06\01	Approved	98\06\01 3.

Change Forecast

1 Scope

1.1 Introduction

This document contains the results obtained from the telephonic interviews conducted during May 1998. These interviews were conducted to find out w. ..the current progress in the adoption of OO in South Africa is.

1.2 Audience

The audience for this document comprise the various stakeholders of the SEAL, including:

- Head of the Department, Electrical Engineering
- Product developer M Jansen van Rensburg
- Product Manager and supervisor Prof Dwolatzky

1.3 Applicable Documents

1.3.1 SEAL QMS Standards

SEAL QMS Document Layout, Presentation and Typesetting Guide, QS 003, Revision 1.00, 3 October 1994

1.3.2 International Standards

a. ISO\IEC 15504 Process Assessment Part 2: A Reference Model For Processes And Process Capability Date: PDTR, October 1996

b. ISO\IEC 9126 (1991) -Information Technology - Software product evaluation - Quality characteristics and guidelines for their use.

1.4 Requirements Traceability

a. ISO 9001 (1994) Clause 4.:

2 The results

The following terminology was used:

N / A	Not applicable
Company size	Number of people in company
IT size	Number of people in IT department
Skill level	Level of training
Project size	Project size given in number of people
Project Lifetime	Average length of projects in company in months
Process	In the process of putting quality procedures in place
< 3	Less than 3
10+	More than 10

Company Name	Location	Company size	IT size	Skill level	OO exposure	Nature of IT business	Quality awareness	Project size	Project Life-time	Uses a methodology
Financial Sector										
Prestasi	National	100-1000	2-10	University / Technikon	Have used OO products	consult	No	2-5	3-6	Almost always
Fedsure	National	1000+	20+	Technikon	Have implemented OO	development	Yes	10+	12+	Never
Southern Life	National	1000+	20+	Self-trained	Have implemented OO	development	Yes	5-10	12+	Never
Momentum Life	National	100-1000	20+	University	Have used advanced OO techniques	development	No	2-5	< 3	Almost always
Future Bank	National	100-1000	11-20	Self-trained	Heard of OO	development	Most depart ments	1	< 3	Never
Unibank	National	100-1000	11-20	University	Have implemented OO	development	no	1	6-12	Never

Company Name	Location	Company size	IT size	Skill level	OO exposure	Nature of IT business	Quality awareness	Project size	Project Life-time	Uses a methodology
Saambou	National	1000+	1	Self-trained	none	support	no	N/A	N/A	N/A
Legal Wise	National	100-1000	2-10	Self-trained	Have implemented OO	support	no	1	< 3	Never
BOE	National	1000+	20+	University	Have used OO products	development	yes	2-5	3-6	Almost never
Huysamer Stals	National	100-1000	2-10	University	Have implemented OO	development	Most depart ments	5-10	6-12	Never
Reserve Bank	Gauteng	1000+	20+	University	Have implemented OO	development	no	2-5	6-12	Almost never
UAL Merchant Bank	National	1000+	20+	all of above	Have implemented OO	development	process	5-10	3-6	Almost always
Wesbank	National	1000+	20+	all of above	none	development	yes	2-5	6-12	Almost always
RMB	National	100-1000	20+	University	Have used advanced OO	development	Most depart	2-5	6-12	Always

Company Name	Location	Company size	IT size	Skill level	OO exposure	Nature of IT business	Quality awareness	Project size	Project Lifetime	Uses a methodology
					techniques		ments			
Stanbic	Gauteng	11-100	1	Self-trained	Have used OO products	support	no	N/A	N/A	N/A
NBS	National	1000+	20+	all of above	Have implemented OO	development	yes	2-5	< 3	Always
Alexander Forbes	National	1000+	20+	University	Have implemented OO	development	yes	varies	12+	Almost always
Liberty	National	1000+	20+	Self-trained	Heard of OO	all of above	no	varies	varies	Almost always
Santam	National	1000+	20+	Self-trained	Have used OO products	development	no	5-10	< 3	Almost always
New Republic Bank	KZN	100-1000	20+	all of above	Have used advanced OO techniques	development	Most depart ments	5-10	3-6	Always
Education Sector										
Radical	KZN	0-10	2-10	Technikon	Have used advanced OO	consult	yes	1	varies	Always

Company Name	Location	Company size	IT size	Skill level	OO exposure	Nature of IT business	Quality awareness	Project size	Project Life-time	Uses a methodology
Training					techniques					
ECP	National	11-100	11-20	Technikon	none	consult	no	2-5	12+	N / A
Birnam	Gauteng	11-100	2-10	Technikon	Have used OO products	consult	no	N / A	N / A	N / A
TSA	National	1000+	11-20	University / Technikon	Have implemented OO	development/ consult	Most departments	2-5	varies	Almost never
Redhill	Gauteng	11-100	2-10	University / Technikon	Have implemented OO	other	no	1	varies	Never
Dept Of Education	National	100-1000	2-10	Technikon	none	support	No	N / A	N / A	N / A
Government Sector										
Spoornet	National	1000+	20+	University / Technikon	Have implemented OO	development	Most departments	5-10	6-12	Always
SABS	Gauteng	1000+	2-10	Self-trained	Have used OO	support	no	2-5	12+	Always

Company Name	Location	Company size	IT size	Skill level	OO exposure	Nature of IT business	Quality awareness	Project size	Project Life-time	Uses a methodology
					products					
Municipality Randburg	Gauteng	1000+	2-10	Self-trained	none	support	Yes	0-1	varies	N/A
SPCA Randburg	Gauteng	0-10	0							
Health Sector										
Adcock	National	1000+	11-20	Self-trained	Have implemented OO	support	No	N/A	N/A	N/A
Wilgers	Gauteng	100-1000	2-10	Self-trained	none	support	no	N/A	N/A	N/A
Medscheme	National	1000+	20+	University	Have used advanced OO techniques	development	process	10+	12+	Always
Kirsch Pharma	Gauteng	0-10	1	Self-trained	none	support	no	N/A	N/A	N/A
Glaxo	National	100-1000	11-20	Self-trained	Have used OO products	development	yes	2-5	3-6	Always

Company Name	Location	Company size	IT size	Skill level	OO exposure	Nature of IT business	Quality awareness	Project size	Project Life-time	Uses a methodology
Brackenfell Hospital	Western Cape	10-100	0							
Hospitality Sector										
SATour	Gauteng	100-1000	1	Self-trained	none	support	no	N/A	N/A	N/A
O'Hagans	National	11-100	1	Self-trained	none	support	no	N/A	N/A	N/A
Southern Sun	National	1000+	1	Self-trained	none	support	no	N/A	N/A	N/A
Seekers	National	100-1000	2-10	Technikon	none	support	no	N/A	N/A	N/A
Imperial	National	100-1000	11-20	Self-trained	Heard of OO	support	no	2-5	3-6	Never
SAB	Gauteng	11-100	2-10	Self-trained	heard of OO	development	no	2-5	6-12	Never
Coaches For Africa	KZN	0-10	0							
IT Sector										
Internet Solution	National	100-1000	20+	University	Have implemented	sales	yes	5-10	varies	Almost always

Company Name	Location	Company size	IT size	Skill level	OO exposure	Nature of IT business	Quality awareness	Project size	Project Lifetime	Uses methodology
					OO					
Computer Foundation	National	100-1000	20+	all of above	Have used advanced OO techniques	all of above	yes	varies	varies	Almost always
SPL	National	100-1000	20+	all of above	Have used advanced OO techniques	development	yes	2-5	6-12	Almost always
QData	National	1000+	20+	all of above	Have used advanced OO techniques	development/ consult	yes	varies	varies	Always
Software Futures	National	11-100	11-20	University	Have used advanced OO techniques	all of above	no	10+	12+	Always
Cenit	National	100-1000	20+	University	Have used advanced OO techniques	development/ consult	yes	10+	1year	Never
Usko	National	100-1000	20+	University	Have used advanced OO techniques	sales	Most depart ments	2-5	12+	Almost always

Company Name	Location	Company size	IT size	Skill level	OO exposure	Nature of IT business	Quality awareness	Project size	Project Lifetime	Uses a methodology
Altech Smartcard	Gauteng	11-100	2-10	all of above	heard of OO	development	no	2-5	3-6	Never
Vircom	Gauteng	100-1000	20+	University	Have used OO products	development	no	5-10	varies	Never
Siemens	National	100-1000	20+	all of above	none	consult	process	5-10	3-6	Almost never
Datatec	National	1000+	11-20	University / Technikon	Have used advanced OO techniques	all of above	yes	5-10	3-6	Almost always
Abstraction	National	11-100	1	University	Have used advanced OO techniques	all of above	yes	2-5	3-6	Almost always
ICL	Gauteng	11-100	2-10	University.	Have implemented OO	development	yes	2-5	6-12	Almost always
Spescom	National	100-1000	11-20	University	Have implemented OO	development	yes	2-5	6-12	Almost never
Brainware	National	1000+	20+	all of above	Have used	all of above	process	varies	varies	Always

Company Name	Location	Company size	IT size	Skill level	OO exposure	Nature of IT business	Quality awareness	Project size	Project Lifetime	Uses a methodology
					advanced OO techniques					
Paradigm	National	100-1000	20+	all of above	Have used advanced OO techniques	all of above	yes	10+	12+	Always
AKS	Gauteng	11-100	20+	University	Have used advanced OO techniques	development	process	10+	12+	Almost always
Energy Measurements	Gauteng	100-1000	2-10	University	Have implemented OO	development	yes	2-5	varies	Almost always
Objectsoft	Gauteng	0-10	2-10	University.	Have used advanced OO techniques	development/ consult	process	2-5	varies	Almost never
Eikon	Gauteng	11-100	11-20	University	Have used advanced OO techniques	all of above	yes	2-5	6-12	Always
Manufacturing Sector										

Company Name	Location	Company size	IT size	Skill level	OO exposure	Nature of IT business	Quality awareness	Project size	Project Life-time	Uses a methodology
BP	National	1000+	20+	University	heard of OO	support	Yes	N/A	N/A	Almost never
Eskom	National	1000+	20+	University	Have implemented OO	development	yes	10+	12+	Always
Temsa	Gauteng	1000+	2-10	Technikon	heard of OO	support	yes	N/A	N/A	N/A
Cargo	Gauteng	100-1000	2-10	Self-trained	none	sales	yes	5-10	varies	Never
Tiger Oats	National	1000+	20+	all of above	none	support	no	N/A	N/A	N/A
Outspan	National	1000+	11-20,	University / Technikon	Have used advanced OO techniques	all of above	yes	2-5	< 3	Always
Plasserall	National	100-1000	2-10	Technikon	Have used OO products	development	no	2-5	12+	Never
VKE	National	100-1000	11-20	Self-trained	Have used OO products	support	no	10+	12+	Never
Nampak	National	1000+	2-10	University	Have used OO products	support	no	2-5	< 3	Never

Company Name	Location	Company size	IT size	Skill level	OO exposure	Nature of IT business	Quality awareness	Project size	Project Life-time	Uses a methodology
Wastetek	National	100-1000	2-10	all of above	none	support	yes	2-5	6-12	Never
ESD	Gauteng	100-1000	2-10	Technikon	none	support	yes	N / A	N / A	N / A
AEC	Gauteng	1000+	20+	all of above	Have used OO products	support	Most departments	N / A	N / A	Never
BMW	National	1000+	20+	University / Technikon	Have implemented OO	all of above	yes	varies	6-12	Always
BIFSA	Gauteng	1000+	0							
Murray Roberts	National	1000+	0							
Kentron	Gauteng	1000+	20+	all of above	Have used advanced OO techniques	development	Most departments	2-5	varies	Always
Mondi-Recycling	National	100-1000	2-10	Self-trained	heard of OO	all of above	no	5-10	12+	Almost always
Dorbyl	National	100-1000	0							

Company Name	Location	Company size	IT size	Skill level	OO exposure	Nature of IT business	Quality awareness	Project size	Project Life-time	Uses a methodology
Marine										
Nanotek	National	100-1000	20+	University	Have implemented OO	development	yes	varies	varies	Never
Polifin	National	1000+	20+	University	none	support	yes	N/A	N/A	N/A
Amalbev	Gauteng / KZN	1000+	20+	Self-trained	Have implemented OO	development	no	2-5	3-6	Almost always
Prieska Engineering	Other	0-10	0							
Burger And Wallace	Western Cape	10-100	0							
Media Sector										
Multichoice	National	1000+	20+	Self-trained	Have used OO products	development	yes	varies	12+	Never
Getaway	National	100-1000	2-10	Self-trained	none	support	no	2-5	varies	Never

Company Name	Location	Company size	IT size	Skill level	OO exposure	Nature of IT business	Quality awareness	Project size	Project Lifetime	Uses a methodology
Computicket	National	100-1000	20+	University / Technikon	Have implemented OO	development	process	5-10	6-12	Almost never
Highveldridge Newspaper	National	1000+	20+	Self-trained	heard of OO	support	yes	5-10	varies	N/A
Primedia Broadcast	National	100-1000	2-10	University / Technikon	Have implemented OO	consult	no	1	6-12	Almost never
Mining Sector										
Mintek	Gauteng	100-1000	2-10	Self-trained	heard of OO	support	process	N/A	N/A	N/A
RBCT	KZN	100-1000	11-20	all of above	Have used OO products	development	yes	5-10	varies	Almost always
RBM	KZN	1000+	20+	Technikon	Have used advanced OO techniques	development	yes	5-10	12+	Almost always
Billiton	National	1000+	2-10	University / Technikon	none	support	yes	N/A	N/A	N/A

Company Name	Location	Company size	IT size	Skill level	OO exposure	Nature of IT business	Quality awareness	Project size	Project Life-time	Uses a methodology
Bastion (Iscor)	National	1000+	20+	University	Have implemented OO	development	no	varies	6-12	Almost always
Samat Mining	National	1000+	1	University	heard of OO	support	no	N/A	N/A	N/A
Retail Sector										
Koljander	Gauteng	11-100	0							
Game	National	1000+	2-10	University	Have implemented OO	development	no	1	< 3	Never
Checkers	National	1000+	20+	all of above	Have used advanced OO techniques	development	no	5-10	12+	Always
Juicy Lucy	National	100-1000	1	University	none	support	no	N/A	N/A	N/A
Edgars	National	1000+	20+	University / Technikon	Have implemented OO	support	yes	10+	6-12	Almost always

Company Name	Location	Company size	IT size	Skill level	OO exposure	Nature of IT business	Quality awareness	Project size	Project Life-time	Uses a methodology
Fotofirst	National	1000+	2-10	Self-trained	Have implemented OO	support	no	2-5	12+	Never
EMI	National	100-1000	2-10	Self-trained	heard of OO	development	Most depart ments	2-5	3-6	Almost always
C N A	National	1000+	20+	University	Have used OO products	consult	yes	5-10	varies	N / A
PicknPay	National	1000+	20+	all of above	Have used advanced OO techniques	development	no	5-10	12+	Always
Angela's Hair Boutique	Gauteng	0-10	0							
Service Sector										
Andersen Consulting	National	1000+	20+	University	Have used advanced OO techniques	development/ consult	yes	10+	6-12	Always

Company Name	Location	Company size	IT size	Skill level	OO exposure	Nature of IT business	Quality awareness	Project size	Project Life-time	Uses a methodology
IMM	National	11-100	2-10	University	Have used OO products	development/ consult	no	1	3-6	Never
Marketel	National	11-100	20+	all of above	none	development/ consult	Most depart ments	2-5	3-6	Never
AOS	National	100-1000	2-10	Self-trained	heard of OO	support	no	varies	< 3	Never
Safmarine	National	100-1000	20+	Technikon	heard of OO	development	yes	5-10	3-6	Always
Twines	Gauteng	11-100	1	Self-trained	none	support	no	N / A	N / A	N / A
Coin Security	National	1000+	1	Self-trained	none	support	no	1	varies	Never
Optimark	Western Cape	11-100	2-10	Self-trained	none	support	no	N / A	N / A	N / A
CPL	National	11-100	20+	University / Technikon	Have used OO products	consult	no	varies	6-12	Almost always
Teljoy	National	1000+	20+	Technikon	none	support	Most depart ments	2-5	12+	Never

Company Name	Location	Company size	IT size	Skill level	OO exposure	Nature of IT business	Quality awareness	Project size	Project Life-time	Uses a methodology
Online Personnel	KZN	0-10	0							



NDM QMS

NDM - Conclusions From Telephonic Interviews

Technical Product

Version 1.00

Document Status: Approved

Table of Contents

Table of Contents ii
Configuration Control iii
Document History iii
Revision History iii
Management Authorisation iii
Change Forecast iii
1 Scope 1
1.1 Introduction 1
1.2 Audience 1
1.3 Applicable Documents 1
1.4 Requirements Traceability 1
1.5 Abbreviations 1
2 Results 2
2.1 The questionnaire 2
2.2 A true representation 2
2.3 Market sector 5
2.4 Skill Level 8
2.5 Quality 11
2.6 Company size 12
2.7 IT Department Size 13
2.8 Project size 15
2.9 Project Life time 16
2.10 Using methodologies 17

Change History

Configuration Control

Project:	NDM QMS
Title:	NDM - Conclusions from Telephonic Interviews
Doc. Reference:	C:\MSC\TP\NDM307.100
Created by:	M Jansen Van Rensburg
Creation Date:	28 June 1998

Document History

Version	Date	Status	Who	Saved as:
0.01	98\07\10	Draft	MVR	NDM307.010
1.00	98\09\23	Approved	MVR	NDM307.100

Revision History

Version	Date	Changes
0.01	98\07\10	Created using NDM002.010
1.00	98\09\23	Approved

Management Authorisation

Version	Date	Status	Project Minute Reference
1.00	98\09\23	Approved	98\09\23 3.

Change Forecast

1 Scope

1.1 Introduction

This document provides some conclusions reached from the telephonic interviews held with 120 companies during May 1998.

1.2 Audience

The audience for this document comprise the various stakeholders of the SEAL, including:

- Head of the Department, Electrical Engineering
- Product developer M Jansen van Rensburg
- Product Manager and supervisor Prof Dwolatzky

1.3 Applicable Documents

1.3.1 SEAL QMS Standards

SEAL QMS Document Layout, Presentation and Typesetting Guide, QS 003, Revision 1.00, 3 October 1994

1.3.2 International Standards

a. ISO\IEC 15504 Process Assessment Part 2: A Reference Model For Processes And Process Capability Date: PDTR, October 1996

b. ISO\IEC 9126 (1991) -Information Technology - Software product evaluation - Quality characteristics and guidelines for their use.

1.4 Requirements Traceability

a. ISO 9001 (1994) Clause 4.2

1.5 Abbreviations

OO Object Orientation

2 Results

2.1 The questionnaire

The questionnaire that was used is given in document NDM302. The questions were chosen with the following objectives in mind:

- To make sure that all types of companies were represented
- To verify the relevance of issues that were found to be important from the literature study
- To explore issues that arose from the first informal interviews

2.2 A true representation

Since the results obtained from the telephonic interview would critically influence the remainder of the research, a representative sample of companies had to be obtained.

Since in South Africa, many companies operate nation wide with offices in various regions, location could not be used as a criterion in the sampling process.(Figure 1)

The criterion that was therefore used was market sector. The number of companies in each sector was selected in proportion to the size of the relevant market sector.

Eleven market sectors were identified:

- financial and banking
- hospitality (including travel, tourism, hotels, restaurants)
- media (including printing and publishing)
- mining
- manufacturing (including engineering, construction and the process industry)

- services
- government
- health
- retail
- education
- IT (including Information Technology, software, telecommunication and electronics)

A list of companies was compiled using several business directories and Internet classification directories as guidelines. The final list of respondents is given in document NDM 301.

Regarding the nature of the IT business in each company, as can be seen in Figure 2, the results obtained from this questionnaire do indeed represent a wide range of companies and therefore the results obtained from this questionnaire do in fact represent the true situation in South Africa:

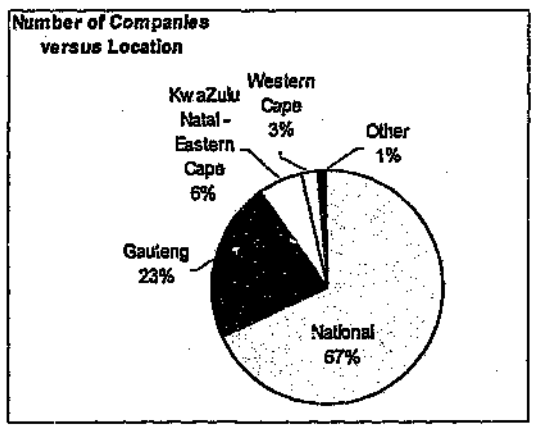


Figure 1 Number of companies versus location

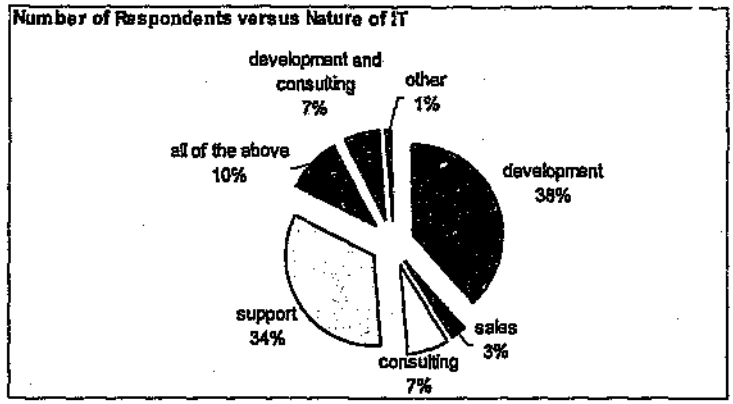


Figure 2 Nature of companies

Figure 3 illustrates the result that was found regarding the presence of OO in the companies studied.

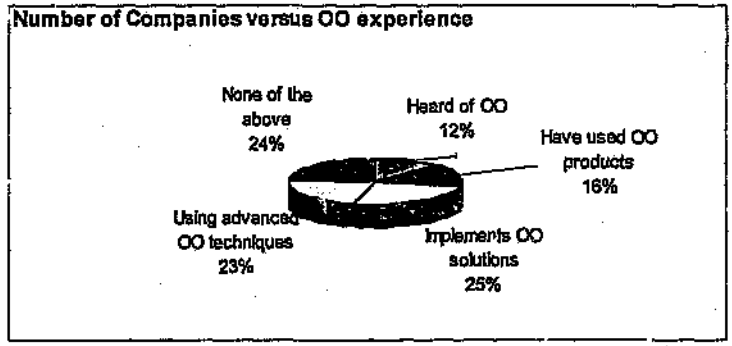


Figure 3 OO experience

The next step is to evaluate specific issues addressed in the research – note that the results that follow include only those companies that possess an IT department. The first graph in each case demonstrates that a representative sample was always taken. Each of the following issues will also be explored in the final interview.

2.3 Market sector

The companies concerned had to be chosen in such a way that the research would represent all market sectors. Figure 4 depicts that this was done.

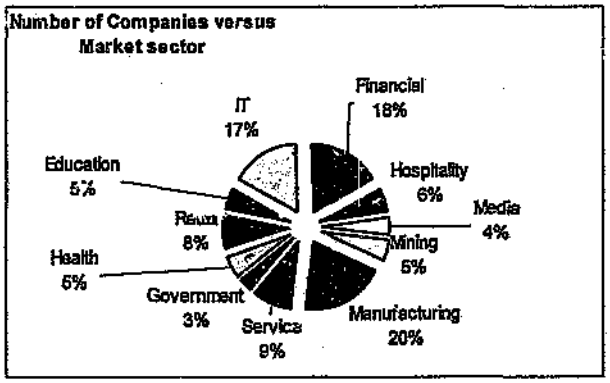


Figure 4 Number of companies versus market sector

Table 2 shows the experience within each market sector, as well as the OO trend within each sector. Data was normalised so that the number of companies interviewed in each sector would not bias the results.

A grey scale was used to indicate the number of companies within a category. A white cell indicates no presence, whereas a dark grey cell indicates a large number of companies in the particular category.

A weighted average was calculated within each market sector, with weights assigned as given in Table 1.

Table 1 Weights

Category	Weight
No OO	1
Heard of OO	2
Have used OO products	3
Have implemented OO	4
Have used advanced OO techniques	5

Table 2 Weighted averages

	Financial	Hospi-tality	Media	Mining	Manufac-turing	Service	Govern-ment	Health	Retail	Edu-cation	IT
No OO	0.100		0.200	0.167					0.125		0.050
Heard of OO	0.100	0.333	0.200	0.333	0.167	0.200	0.000	0.000	0.125	0.000	0.050
Have used OO products	0.200	0.000	0.200	0.167	0.222	0.200		0.200	0.125	0.167	0.050
Have implemented OO		0.000		0.167	0.222	0.000		0.200			0.200
Have used advanced OO	0.150	0.000	0.000	0.167	0.111	0.100	0.000	0.200	0.250	0.167	

Weighted Average	3.45	1.33	2.8	2.83	2.89	2.1	2.7	2.8	3.5	3	4.35

Looking at the three highest averages obtained (also given in Table 2) it would appear that the financial, retail and IT sectors have progressed further in the adoption of OO.

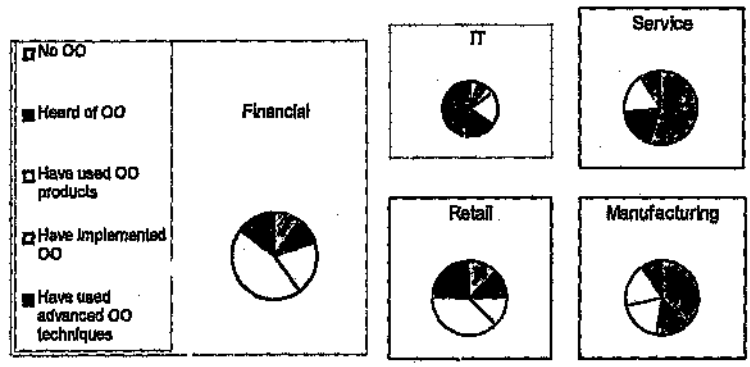


Figure 5 OO experience in various market sectors

Figure 5 illustrates the OO experience in various market sectors: the retail, financial and IT sectors are ahead, while the manufacturing and service sectors are lagging in OO experience.

2.4 Skill Level

Figure 6 illustrates the level of training of employees in the companies interviewed.

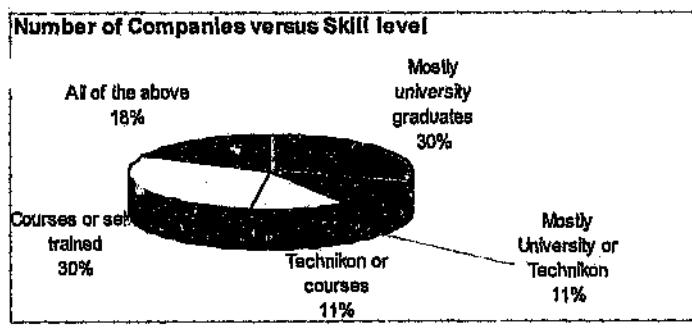


Figure 6 Number of companies versus skill level

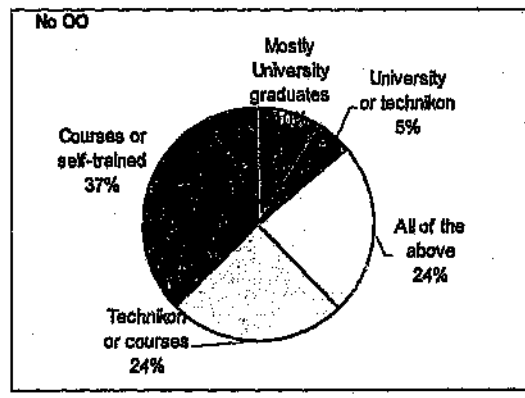


Figure 7 Skill level in companies with no OO experience

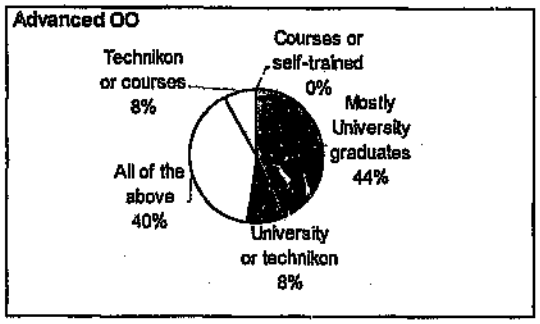


Figure 8 Skill level in companies using advanced OO techniques

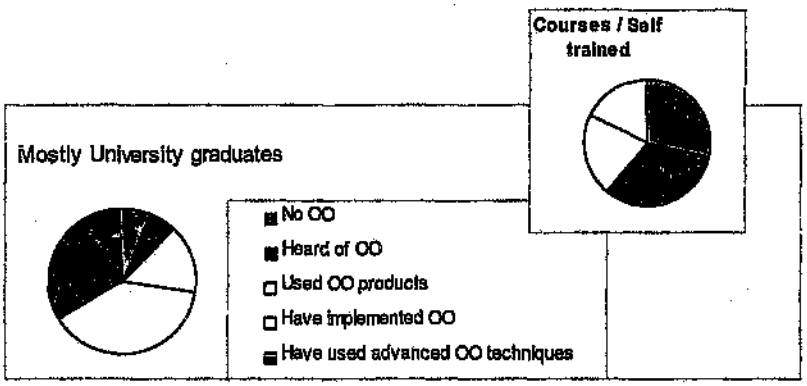


Figure 9 Comparing OO experience

Figures 7, 8, and 9 illustrate the relationship between the skills in the companies and the experience in OO.

The results show a clear relationship between the skill level of the employees involved and the progress thus far in the implementation of OO. The question that therefore arose was: Is formal academical education a prerequisite for the successful implementation of OO?

2.5 Quality

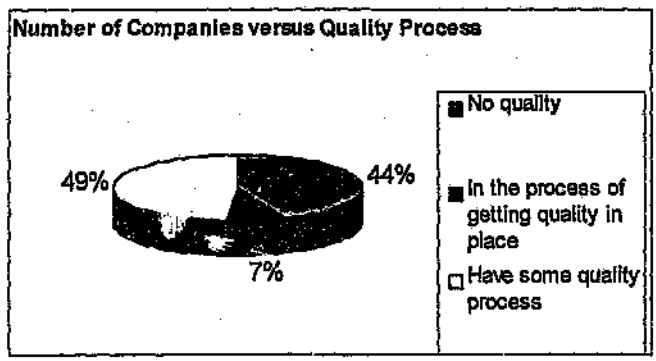


Figure 10 Number of companies versus quality process

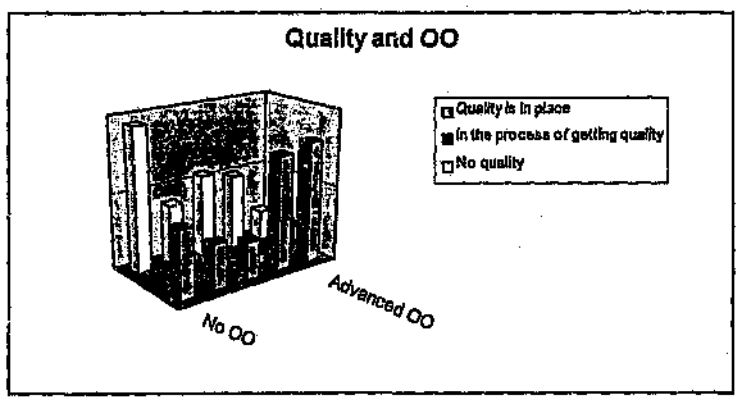


Figure 11 Quality and OO

Figure 10 illustrates the presence of quality processes within the companies interviewed. Figure 11 illustrates the relationship between quality in these companies and the awareness of OO. Results indicate a clear relationship between quality in the company and the implementation of OO. Where quality processes were present, the adoption of OO was more advanced.(Figure 12).

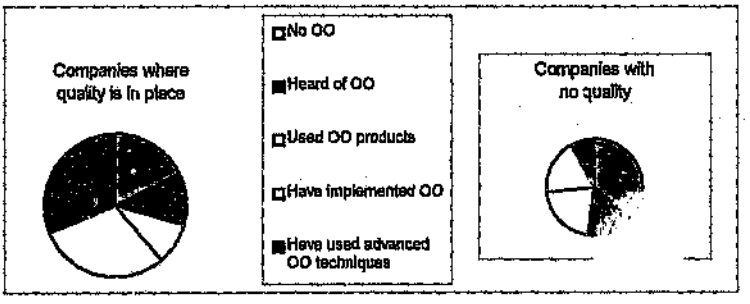


Figure 12

Figure 12 further illustrates this relationship from a different perspective.

This lead to important questions that would be answered in the final interview, namely is there a correlation between OO and quality? If so, does OO lead to the improvement of quality in the company, or does quality lead to a more successful implementation of OO?

2.6 Company size

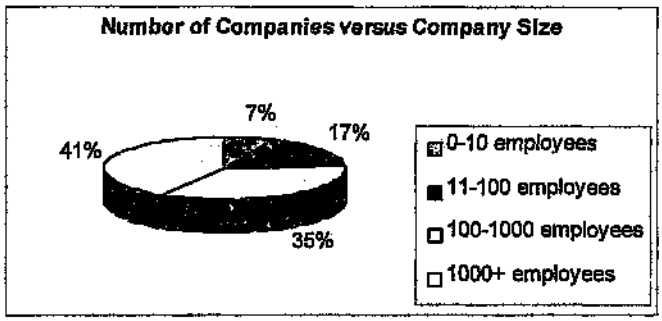


Figure 13 Number of companies versus company size

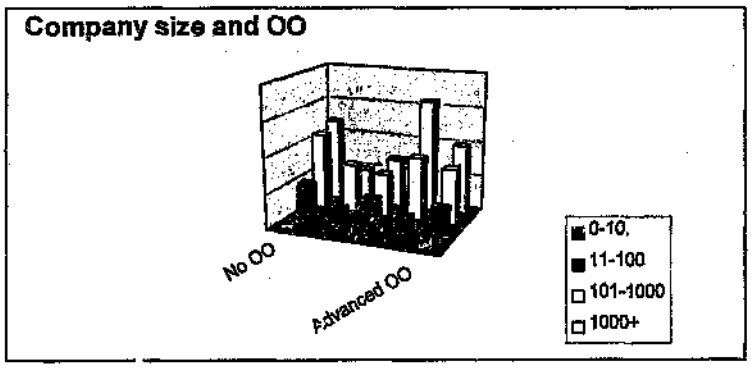


Figure 14 Company size and OO

Figure 13 illustrates the representation of companies of various sizes during the telephonic interviews. Figure 14 shows that there is little or no relationship between the company size and the transition towards OO.

2.7 IT Department Size

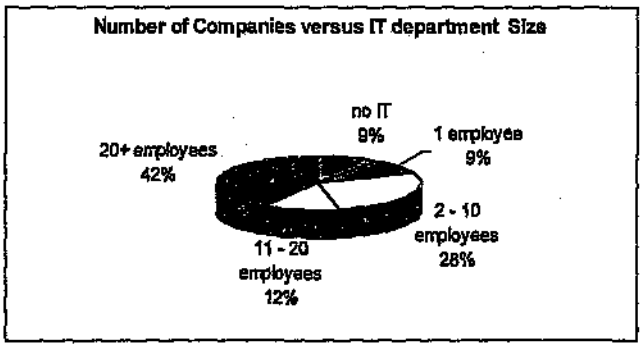


Figure 15 Number of companies versus IT department size

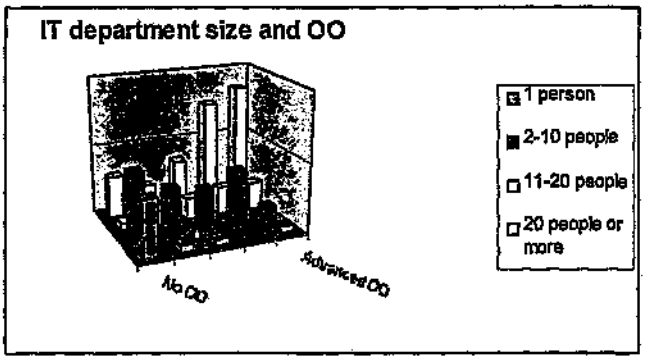


Figure 16 IT department size and OO

Figure 15 illustrates the size of the IT departments found in the companies interviewed.

Trends in Figure 16 show a clear relationship within each OO experience sector but also across sectors: the larger IT departments seem to be more advanced in moving towards OO.

Is there therefore an optimal IT department size when moving to OO? This issue would be investigated in the final interviews.

2.8 Project size

The typical project size that could be found in the companies interviewed is illustrated in Figure 17.

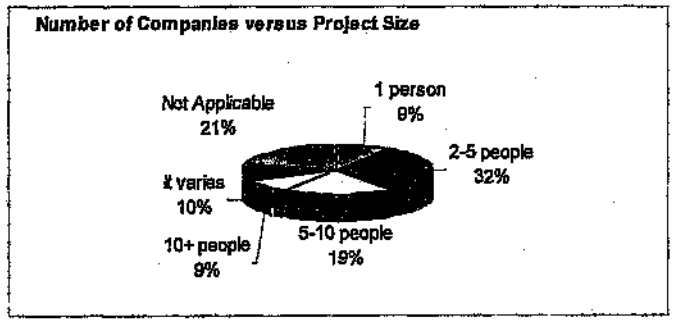


Figure 17 Number of companies versus project size

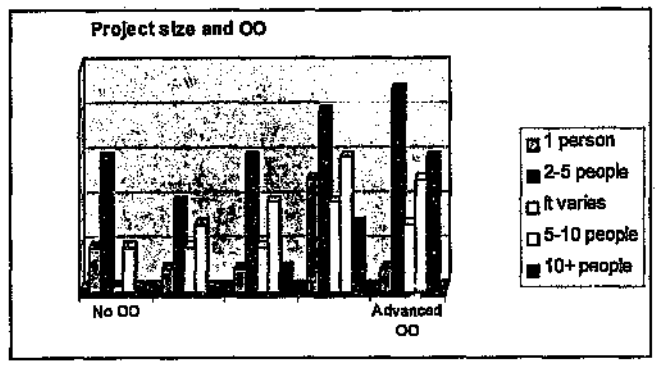


Figure 18 Project size and OO

From Figure 18, it does not appear as though there is any relationship between project size and the successful implementation of OO.

2.9 Project Life time

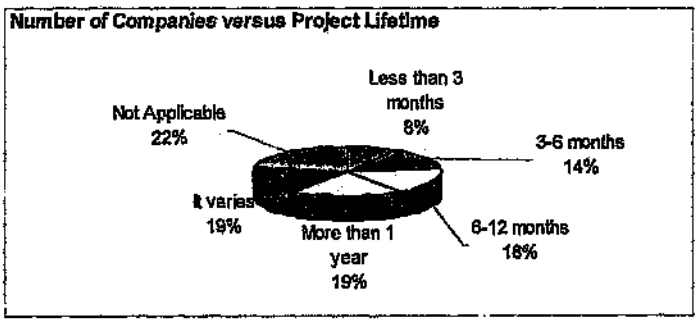


Figure 19 Number of companies versus Project Lifetime

Figure 19 illustrates the typical project lifetime in the companies interviewed.

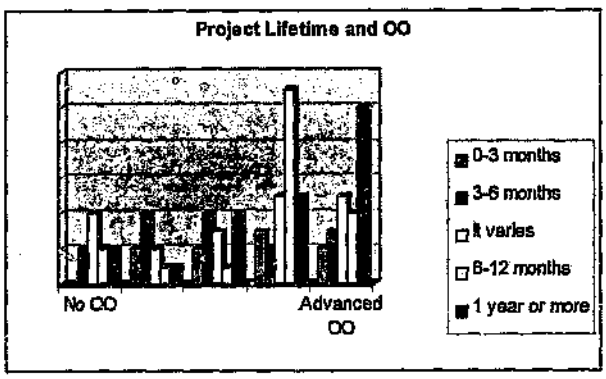


Figure 20 Project Lifetime and OO

Figure 20 illustrates that no significant conclusions could be drawn from the results obtained regarding project lifetime. This factor (i.e. no relevance) would therefore also be included in the final interview.

2.10 Using methodologies

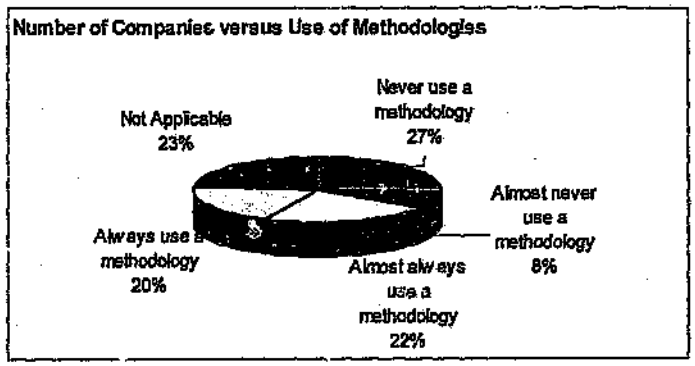


Figure 21 Number of companies versus use of methodologies

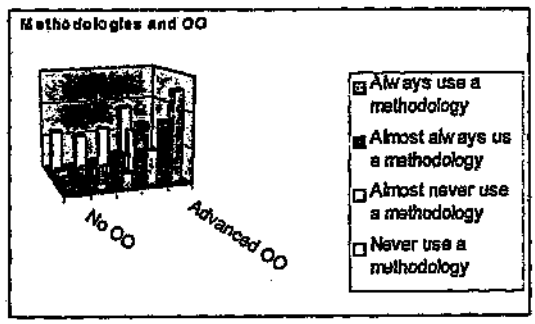


Figure 22 Methodologies and OO

Figure 21 illustrates the presence of methodologies in the companies interviewed.

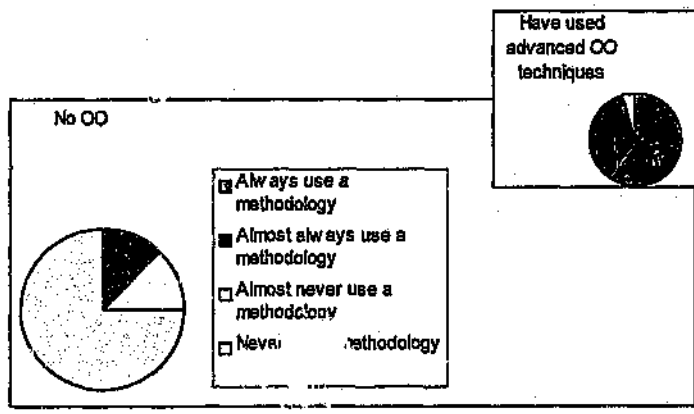


Figure 23 The relevance of using methodologies

There is a clear relationship between the presence of a methodology and the awareness of OO within companies. In the case where a methodology is used, the adoption of OO is mostly in an advanced stage already. Similarly, where no methodology is used there has also not been much progress towards the adoption of OO. (Figures 22,23)

■



NDM QMS

NDM - Local Interview Questionnaire

Technical Product

Version 1.00

Document Status: Approved

Table of Contents

Table of Contents iv

Change History v

Configuration Control v

Document History v

Revision History v

Management Authorisation v

Change Forecast v

1 Scope 6

1.1 Introduction 6

1.2 Audience 6

1.3 Applicable Documents 6

1.4 Requirements Traceability 6

1.5 Abbreviations 7

2 Introduction 8

3 The facts 9

3.1 The company 9

3.2 Why did the company adopt OO? 9

3.3 Why not OO 9

3.4 Time 10

3.5 Is OO still emerging? 10

3.6 First project 11

3.7 Previous experience 11

3.8 Cultural change 11

3.9 Language 11
3.10 Reuse 12
3.11 Reengineering legacy systems 12
3.12 Testing 12
3.13 Metrics 13
3.14 Project statistics 13
3.15 Mistakes 13
4 Opinions 14
4.1 Location 14
4.2 Time 14
4.3 Risks involved 15
4.4 Management 15
4.5 Paradigm shift 15
4.6 Company profile 16
4.7 Skill level versus OO experience 16
4.8 Organisational issues 18
4.9 Quality versus OO experience 18
4.10 Usage of methododologies versus OO experience 19
4.11 Data storage 20
4.12 Project profile 20
4.13 Patterns 21
4.14 Physical environment 21
4.15 What's next for OO 21
4.16 Year 2000 22
4.17 Services available in SA to help 22
4.18 Global situation vs SA companies 22

4.19 Guidelines 22

Change History

Configuration Control

Project:	NDM QMS
Title:	NDM - Local Interview Questionnaire
Doc. Reference:	C:\MSC\TP\NDM304.100
Created by:	M Jansen Van Rensburg
Creation Date:	2 June 1998

Document History

Version	Date	Status	Who	Saved as:
0.01	98\06\02	Draft	MVR	NDM304.010
1.00	98\06\08	Approved	MVR	NDM304.100

Revision History

Version	Date	Changes
0.01	98\06\02	Created using NDM002.010
1.00	98\06\08	Approved

Management Authorisation

Version	Date	Status	Project Minute Reference
1.00	98\06\08	Approved	98\06\08 3.

Change Forecast

1 Scope

1.1 Introduction

This document provides a record of the questionnaire used during local interviews conducted in 12 South African companies during June and July 1998.

1.2 Audience

The audience for this document comprise the various stakeholders of the SEAL, including:

- Head of the Department, Electrical Engineering
- Product developer M Jansen van Rensburg
- Product Manager and supervisor Prof Dwolatzky

1.3 Applicable Documents

1.3.1 SEAL QMS Standards

SEAL QMS Document Layout, Presentation and Typesetting Guide, QS 003, Revision 1.00, 3 October 1994

1.3.2 International Standards

a. ISO\IEC 15504 Process Assessment Part 2: A Reference Model For Processes And Process Capability Date: PDTR, October 1996

b. ISO\IEC 9126 (1991) -Information Technology - Software product evaluation - Quality characteristics and guidelines for their use.

1.4 Requirements Traceability

a. ISO 9001 (1994) Clause 4.2

1.5 Abbreviations

OO Object Orientation

2 Introduction

The rest of this document contains the questionnaire that was used during interviews conducted with 12 selected South African companies. These interviews were conducted in person. The duration of each interview was approximately one hour. Text in italics provides the respondent with additional information to help explain the issue at hand. The questionnaire probed into the progress that companies have made in the transition to Object Orientation.

3 The facts

3.1 The company

The company size:

The IT department: (also, how many contractors?)

Can you briefly describe the activities in the IT department?

3.2 Why did the company adopt OO?

(For example, was it for encapsulation, reuse, flexibility? Was it for higher quality systems? To help the development process? Large systems? Did you find previous methods lacking?)

(Page-Jones[76] talks about right and wrong reasons for moving to OO. The wrong reasons are: competitors are using it, using C++ purely as a better C, falling for the so-called "Fad of the year")

3.3 Why not OO

If no transition to OO was done, why not?

Did the company use other methods? Which methods and why?

3.4 Time

When did the company start getting involved in OO?

Did the time when the transition was done play a role? If so, what role?

What factors influenced the speed of adopting OO in the company?

3.5 Is OO still emerging?

Is OO still an emerging technology or has the technology matured?

("The characteristics of an emerging technology according to Ken Orr are that there is more written about it than known about it, there are more people selling it than using it, and the vendors are making more money from education than from selling the tools" These things still seem to apply to OO." –Pancake[78])

3.6 First project

How did you approach the first OO project? Any specific characteristics?

(Several articles mention budget, having a lenient schedule, high visibility, that it should be a new project without legacy issues, that it should be a large meaningful project?)

3.7 Previous experience

What were the previous work methods? Did you use a structured methodology or perhaps no methodology?

3.8 Cultural change

Did the company have to deal with any resistance to OO? How did you deal with that?

3.9 Language

Which language did you use and why?

3.10 Reuse

Any Problems?

(Several articles talk about over- and under- generalisation. Also, many articles complain about developers still being rewarded for writing new code instead of being rewarded for reuse)

Any Guidelines?

How does your company manage the retrieval of reusable objects? Do you have some kind of librarian who knows which objects exist?

3.11 Reengineering legacy systems

Any Problems?

Any Guidelines?

(For example, some articles propose using wrappers)

3.12 Testing

Did you have any problems or suggestions?

("Ian Graham[37] recommends re-testing all subclasses when modifying a super class and similarly, when altering a subclass, re-testing the inherited operations as well as the new ones. One way to test new subclasses is to test the super class, test the subclass

with stubs substituted for inherited features and finally test the two with the stubs removed. ")

3.13 Metrics

In an article by Korson[47], he mentions that "approaching management purely with technical aspects will not succeed - one has to demonstrate improved productivity."

What metrics did you use for tracking progress in the move towards OO? How did these metrics rate? For example: What do you use for data collection of statistics, time spent on projects, etc?

3.14 Project statistics

Can you perhaps give me any statistics in your transition to OO? For example, what costs were involved, etc.?

3.15 Mistakes

What mistakes did your company make when moving to OO? Are there any problems at present?

4 Opinions

4.1 Location

Do you think that there is a significant difference in the use of OO in Cape Town versus Johannesburg and Kwazulu Natal? If so, is this difference skills related?

4.2 Time

What in your opinion is the likely time frame a company needs for adopting OO?

It seems as if the software industry has improved a lot - would an organisation beginning with OO technology now have a faster program? Especially since the OMT has now established certain standards.

(Some articles however also mention that there are lots of tools available now which can lead to confusion about what is right.)

Do you perhaps know how the progress of OO here in South Africa compares with the global situation?

4.3 Risks involved

How do you think should the risks and costs involved in moving to OO be dealt with?

4.4 Management

Do you think OOT insertion is best achieved top down or bottom up?

Do you think there is any difference in the management of OO projects and normal management?

Do you perhaps have any guidelines for the management of OO projects?

4.5 Paradigm shift

Do you think that, when moving from C to C++, it is practical to start a new project using C++ simply as a better C and then move

to OO techniques, or adopt the new paradigm and language from the start? In other words, is a revolutionary (shock therapy) approach preferable to a more gradualist evolutionary approach?

4.6 Company profile

Regarding company size, do you think that the progress towards OO in a company is related to the size of the company? What is the reason?

Regarding IT department size it seems as if larger departments are doing better in the adoption of OO (directly equivalent to OO) whereas the progress in smaller departments is slower. Would you agree? Why?

In your opinion does the market sector in which the company falls play a role in the move towards OO? Are there significant differences? It seems as though the retail, IT and financial sectors are ahead in the adoption of OO.

4.7 Skill level versus OO experience

Regarding OO training – do you have any suggestions?

What problems did you experience?

Articles stress the importance of training management as well - what is your opinion?

What is the state of the training organisations available? How do they rate?

Regarding staffing – do you have any suggestions?

(Litvintchouk[68] said that the most important factor in getting OO technology inserted, was skill leverage....the good procedural designers became the good object designers, probably because they had abstract reasoning capability)

Do you have any staffing problems?

From the telephonic questionnaire it seems as though companies employing mostly university-trained people are more advanced in the adoption of OO. Related to this, it seems as if companies with mostly self-trained employees are less advanced in OO. Would you agree with these findings? Why?

4.8 Organisational issues

In your opinion, are there any organisational issues that play a role in the move towards OO? Did the organisational structure change in your case, for example?

4.9 Quality versus OO experience

Did the pressure to get ISO 9000 play a role in moving towards OO?

Does the SEI (CMM) level play a role at all?

(Some authors feel that the company should be at level 3 to maximise reuse, that the transition can only be as orderly as the SEI level)

From the telephonic interviews it seems as though quality awareness is directly equivalent to OO experience in companies and related to that, where there is no quality there is also not any OO experience.

Do you use configuration management?

4.10 Usage of methodologies versus OO experience

From the telephonic interview it seems as if there is a direct equivalent relationship between the use of methodologies and OO experience. Do you agree?

Regarding CASE tools, which tools do you use and which criteria did you use for choosing the tool?

(Articles mention support for graphical notation, code generation, browsing capabilities, error checking, group ware support, etc.)

Are there still any problems involved in the CASE tools available?

Which methodology do you use?

(For example, UML, Booch, Coad/Yourdon, Shlaer/Mellor, OOSE/Jacobson)

Which levels of the System Development Life Cycle do you use methodologies and tools for? For example analysis, design, testing etc?

In your opinion, what is the state of the methodologies and tools available?

4.11 Data storage

According to the Cutter Consortium Report of 1997, smaller companies have progressed towards OO databases faster than large companies? Do you agree? What was your experience?

"Large companies are generally ahead of smaller companies in adopting OO. Large companies are likely to have CORBA and C++/Java while small companies are ahead in adopting OO databases, reflecting the stronger hold that database managers in large companies have on their organisations."[38]

4.12 Project profile

Regarding the project size – do you think this plays a role in the transition towards OO?

(In 1994, Hartman[41] *reported that most current OO methodologies and tools do not yet provide a suitable level of coverage and support for the construction of large scale systems)*

Re · ·ding the project lifetime, from the telephonic interview it does not seem as though the lifetime (duration) of the project development plays any role in the adoption of OO. Do you agree?

4.13 Patterns

Regarding design patterns, did you have any problems or do you have any guidelines?

4.14 Physical environment

Do you think that the physical environment plays a role ? For example should developers sit together in the same office / group ?

4.15 What's next for OO

What do you think will be the next developments in OO?

4.16 Year 2000

Does the Y2K issue have an influence at present, in terms of perhaps employees used to solve Y2K problems and taken away from new OO development?

4.17 Services available in SA to help

Are you aware or using any services available in South Africa that can assist companies in the move to OO?

4.18 Global situation vs SA companies

What differences do you see in the global transition towards OO versus South Africa?

4.19 Guidelines

Do you have any other guidelines for companies contemplating the move to OO?



NDM QMS

NDM - The Local Interviews: Conclusions

Technical Product

Version 1.00

Document Status: Approved

Table of Contents

Table of Contents ii

Change History iv

Configuration Control iv

Document History iv

Revision History iv

Management Authorisation iv

Change Forecast iv

1 Scope 1

1.1 Introduction 1

1.2 Audience 1

1.3 Applicable Documents 1

1.4 Requirements Traceability 1

1.5 Abbreviations 1

2 Results from the Questionnaire 2

2.1 The companies 2

2.2 Does the time when moving to OO have any influence on the success of the adoption? 6

2.3 How long will it take? 7

2.4 Reasons for moving to OO 7

2.5 Previous methods used 8

2.6 Resistance 8

2.7 Reuse 10

2.8 Legacy systems 11

2.9 Testing 12

2.10 Quality 13
2.11 Location 14
2.12 First project 15
2.13 Is OO still emerging or has the technology matured? 15
2.14 Physical area of work 17
2.15 The Year 2000 crisis 17
2.16 OO insertion: Bottom up or top down? 18
2.17 A difference in management 18
2.18 Evolution or Revolution? 19
2.19 Training 20
2.20 The Global situation 22
2.21 Companies providing an OO service 24
2.22 Databases 24
2.23 Company profile 25
2.24 Market sector 27
2.25 Methodologies 28
2.26 CASE tools 29
2.27 Metrics 30
2.28 Organisational structure 31
2.29 Speed of adoption 32
Language 33
2.31 Mistakes 34
2.32 Guidelines 35
2.33 A comparison with Safmarine 37

Change History

Configuration Control

Project:	NDM QMS
Title:	NDM - The local interviews: conclusions
Doc. Reference:	C:\MSC\TP\NDM310.100
Created by:	M Jansen Van Rensburg
Creation Date:	28 June 1998

Document History

Version	Date	Status	Who	Saved as:
0.01	98\07\10	Draft	MVR	NDM310.010
1.00	98\09\23	Approved	MVR	NDM310.100

Revision History

Version	Date	Changes
0.01	98\07\10	Created using NDM002.010
1.00	98\09\23	Approved

Management Authorisation

Version	Date	Status	Project Minute Reference
1.00	98\09\23	Approved	98\09\23 3.

Change Forecast

1 Scope

1.1 Introduction

This document provides conclusions reached during the local (final) interviews held with representatives from 12 South African companies during June and July 1998.

1.2 Audience

The audience for this document comprise the various stakeholders of the SEAL, including:

- Head of the Department, Electrical Engineering
- Product developer M Jansen van Rensburg
- Product Manager and supervisor Prof Dwolatzky

1.3 Applicable Documents

1.3.1 SEAL QMS Standards

SEAL QMS Document Layout, Presentation and Typesetting Guide, QS 003, Revision 1.00, 3 October 1994

1.3.2 International Standards

a. ISO\IEC 15504 Process Assessment Part 2: A Reference Model For Processes And Process Capability Date: PDTR, October 1996

b. ISO\IEC 9126 (1991) -Information Technology - Software product evaluation - Quality characteristics and guidelines for their use.

1.4 Requirements Traceability

a. ISO 9001 (1994) Clause 4.2

1.5 Abbreviations

OO Object Orientation

2 Results from the Questionnaire

The interviews were held in person with 12 South African companies selected from the original 120 that were used for the telephonic interviews. The duration of these interviews was approximately one hour each. Questions were formulated as issues for discussion rather than simple yes no or multiple choice questions. The questionnaire that was used is given in document NDM304. Questions came from observations in the results from the telephonic interviews, the earlier discussions during informal interviews and the literature study.

NOTE: Where a statement refers to a specific company's opinion or experience, the relevant company's name appears in brackets at the end of the statement.

2.1 The companies

The following companies were interviewed:

2.1.1 Computer Network Services, University of the Witwatersrand (CNS)

The department consists of 75 people of which 20 to 25 people do development. The department is responsible for the university's computing services and consist of 3 main areas, which are facilities (maintenance of machines), ITS (support, packages, users, Novell administration) and applications (development on the administration systems).

They are currently busy with their first OO project, which started at the beginning of 1998. The project, a generic marking system, will be done in Java and they have decided that if making a transition to OO, the transition should be done full blown.

The prompt to start the project was the fact that the existing system is not Year 2000 compliant. The project is extremely critical. Current challenges include the need to train people and the learning curve involved.

2.1.2 Rand Merchant Bank (RMB)

The company, classified for the research as a financial sector company, consist of approximately 500 people, and there are about 30 people in the IT department. There are 2 main areas within the department – support (hardware) and development

The company started development in OO in 1990 after similar work was done in Australia. The first project took 1 year to develop.

2.1.3 Company A

The company, a medium sized company in the electricity management sector consists of 70 people. The IT department consist of 10 people. The company develops electricity management systems which involves PC based applications as well as embedded software. Informally the company has been involved in OO for 5 years but formal OO processes as a group have only started recently.

2.1.4 EIKON

The company consists of 15 people of which 11 fill technical positions. There are two areas: services (mentoring, training and development) and product distribution.

The company was started as an ICL venture in 1993, after being awarded the Spoornet tender for OO development.

EIKON also has a close relationship with the Swedish company Objectory, which provides a link with the OO guru Ivar Jacobson.

Also included in their services is the performance of assessments of companies who have started OO by themselves and who now want to know if they are on the right track in the transition.

2.1.5 Outspan

The company consists of over 1000 employees in South Africa and also has various sister companies overseas.

There are 15 people in the IT department. Activities include system development of financial systems and logistics, as well as support of existing systems. Involvement in OO has started 2 years ago.

2.1.6 Company B

The company, a medium sized company in the defence industry, consists of over 1000 people, and there are 200 in IT development. There are two main areas – system analysis and development and SAP. On the system analysis side they have been using OO intensively for 2 years

2.1.7 Company C

The company, a small technical software development house, was started 12 years ago and develops, markets and sells software for the telecommunications industry. The focus is on engineering and technical software. The company consists of approximately 20 people and has been involved in OO now for 3 years.

2.1.8 ObjectSoft

The company consists of less than 10 people, and has been in existence for approximately one year. Functions include providing solutions for client server applications as well as Internet solutions.

Since the company is new, there are no legacy systems involved and therefore also no Year 2000 problems.

2.1.9 Momentum Life

The company, classified for the research as being financial, consist of less than a thousand people and have 70 people in the IT department

Activities within the IT department include the usage of midrange computers (AS400) and PCs for insurance administration and endorsements systems. Involvement in OO started 3 years ago.

2.1.10 Safmarine

In this case, the company has not been involved with OO, but has just made a very successful transition to information engineering. It was felt that it could therefore be useful to see what there is to learn from the transition that was done in this case.

The company consists of less than 1000 people and would like themselves to be classified as a medium sized company. The IT department consist of more than 20 people and is responsible for support and development (if the necessary resources are available, otherwise outsourcing takes place).

The project under discussion, called the Quay Project, was recently nominated for the Computer World Smithsonian Core Award. The project was an extremely critical project, also because a transition took place from mainframe to UNIX.

The company started redeveloping all systems in 1990, after doing a complete research of the technologies available at the time. It was felt that OO was immature at the time whereas the Information Engineering tools were already developed. Previous methods used included COBOL programming.

2.1.11 PicknPay

This retail company consist of more than 1000 people and has 75 people in the IT department.

Applications development includes back office systems (COBOL-based), point of sale systems and distribution.

The transition to OO was initiated 2 years ago when a change in database became necessary – at the time it was decided that the architecture should also be changed. A motivation for OO was that, although the first project would be more expensive, in the long term it would be beneficial.

2.1.12 Santam

This financial company consists of over 1000 people. There are 175 people in the IT department. Activities include the development of short term insurance and underwriting applications. Previous systems were mainframe based. The new systems are being developed using Visual Basic and Sybase. Employees see themselves as having only really moved to client-server and starting with OO now.

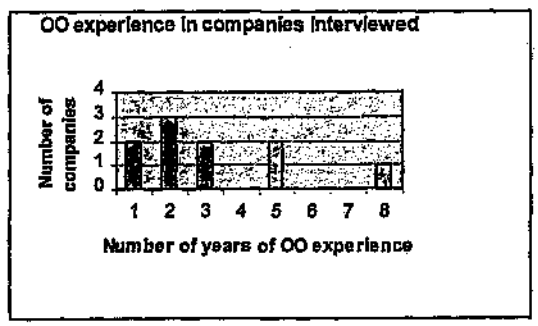


Figure 1 OO experience in companies interviewed

The experience in these companies therefore ranges from less than one year of OO experience to eight years of experience.(Figure 1)

The responses to the questions asked during the interviews were consolidated. Each question will now be discussed in more detail.

2.2 Does the time when moving to OO have any influence on the success of the adoption?

There were mixed feelings about the question whether it would be easier if one would start implementing OO now rather than in the past. Arguments for the statement were:

- there are formal methods and training available. (RMB)
- Object Technology is accepted now.(RMB)
- more development tools are now available (Company A)
- OO has not matured yet but has become practical to implement.(Company A)
- the technology has been proven (Outspan)
- it is best to start using OO now because the hype is over and the facts are on the table – especially for the companies who are not bleeding edge. (Santam)

Arguments against the statement:

- if there are existing skills from the old systems (structured methods) it is going to take longer, also if the company has investments in the old technologies.(Outspan)
- it will not be easier now because tools will not solve the problems. (Company B)
- if starting now, having the knowledge that a company has now after having moved to OO it will be easier, but for a company starting now for the first time it will not be any easier – having new tools will not change that – they will make the same mistakes. (Company C)

It was however also mentioned that the longer a company waits the longer it will typically sit with old technology, which was paid for at a time when they should have already moved to OO. (Outspan) Also, most of the new development tools on the market are based on OO which almost forces companies to join.

2.3 How long will it take?

The time that a company will spend to make the transition to OO was estimated from 6 months (according to EIKON) to 3 years (Outspan). However the following factors were mentioned that should be taken into account when estimating:

- the 3 AHAs: first there is the language, then learning about objects, and lastly the paradigm shift and the integration of it all. It is this last phase that takes the longest. (RMB)

- the company size: for 5 people learning the new concepts it might be a quick process, compared to a large company changing its way of thinking and developing.

- if there is any old data involved it will take a lot longer than for companies where new development is started

- company politics

2.4 Reasons for moving to OO

The conventional reasons mentioned for choosing OO were as follows:

- handling complexity (RMB, Company B)

- the promise of portability (CNS)

- maintainability (Company A, Outspan). EIKON explained about so-called "rest in peace" (RIP) code and how less than 5% of code doesn't need changing which is why OO should be used.

- reuse (Company A, CNS)

- improving quality (Company A)

- understanding the business (Outspan)

- a need to model data and functions together (Company B)

- long term benefits (PicknPay) – the first project will probably take longer but after that the time saving will start.

Surprisingly, reuse was very low on the list of most companies. Therefore this issue will also be discussed separately.

Some interesting (surprising) reasons for moving to OO that were mentioned numerous times were the following:

- All the new technologies are in OO (Santam) There is a need to move with the new technologies available (CNS)
- Developers' needs – developers do not want to work on mainframe systems. Companies (CNS) want to keep the right people and are therefore forced to move to the new technologies.
- Users' needs – as with developers, users do not want to work on mainframe systems, but rather on PC based applications. (Santam)(CNS)

2.5 Previous methods used

In all the cases studied the previous methods that were used were structured methods, with languages being mostly COBOL and Natural Adabas. In most cases there was no real design, it was more a case of just "coding".

2.6 Resistance

A clear picture was forming when this question was asked: without exception, in the companies where OO was implemented with a clear understanding of the paradigm shift and the new design methods, (i.e. full OO) the process was met with a lot of resistance. The opposite was also true: in the companies where people thought they were implementing OO, but were actually just using the new tools such as Delphi: people seemed to be very keen on the idea of OO and therefore were not resistant.

The resistance seemed to stem from the following:

- It seemed to be the older developers who resisted the move to OO (RMB, Outspan, PicknPay)
- Often being uninformed, COBOL developers thought of OO as vapourware (a lot of hype but no results).

- Developers didn't understand the technology and thought it was too complex – even now this is still the case. The developers are often business people who do not see the need for change.

- Uninformed users (RMB)

- Management being uninformed or worried about costs (Company B, PicknPay)

Where there was no resistance the following were possible causes:

- In one case (Company A) everyone was keen because they knew the technology, which again implies appointing people who are open to change.

- an innocence, where people are not really aware of what exactly the new process entails

According to EIKON the resistance should be dealt with as follows. (Figure 2)

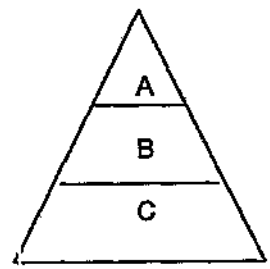


Figure 2 Resistance towards OO

The top area signifies the young open-minded developers. The bottom area signifies the old developers who will not change . The middle area are people in-between who can be convinced either way.

The recommendation is that perhaps this is the area to concentrate on since many developers fall in this "in-between" area.

2.7 Reuse

As mentioned before it became clear that reuse was not a primary objective for many of the companies. Reasons were:

- It is difficult (RMB, Outspan) EIKON mentioned the example of defining a generic customer class where people will argue ad infinitum about what the class should look like. The recommendation is therefore: rather promote maintainability, risk management, etc.
- Reuse is tied to the kind of business and is therefore often too specific rather than too generic[1]
- Where the software is embedded and the hardware changing it is impractical to try to reuse.
- Many of the reusable classes can be obtained from the tool being used

Guidelines

- rather use refactoring[2] afterwards(RMB)

[1] This is also referred to in Korson's article[47].

[2] Often when changing code, additions become very complex but there is no time for redesign. Refactoring describes the techniques that reduce the pain of redesigning. The functionality of the software is not changed, rather the internal structure. It normally involves small steps at a time, such as moving a field to another class. Also, while refactoring no new functionality is added. The technique is still new and is mainly used in the Smalltalk community but promises to improve software development in all environments.[21]

- it is better to rather write a framework and making that specific (Outspan)
- rather concentrate on having the same development process throughout (EIKON)
- reuse comes from repetition where one goes through multiple abstractions - not up front in the design. Trying to design for reuse is therefore considered premature. Concentrate on use rather than reuse (Momentum)

Most of the companies are however investigating having one person being a "reuse miner" / librarian to manage the libraries, plan the repository, etc.

2.8 Legacy systems

Almost all the companies had systems they have recognised as legacy systems. Interestingly, in quite a few cases these systems are relatively new (3 years), and some are even OO systems.[3]

Outspan describes their problem as being twofold – there is historical data which previous people worked with and that must now be used by new people, and secondly, there is a large quantity of old information.

Guidelines for handling the legacy systems can be found in these companies' methods of handling the problem:

- Map the system with a CASE tool, and use messaging and CORBA to wrap it.(EIKON)
- use integration, encapsulation, and queues to integrate at the most granular level in the case of AS400 systems where there is data entry in multiple places.(Momentum)
- file integration
- using data conversion where data is the only contact between the old and new systems.

[3] Casais also referred to this problem in his article[15]

- replacing old modules with new modules as they are being rewritten
- ideally one should rebuild these systems but practically it is often impossible. The suggestion given was hoping that the old systems for users using old operating systems will die out soon

2.9 Testing

During some of the interviews it appeared as if testing did not receive much attention (automated testing was not present) even though the comment was made during the PicknPay interview that testing is the most important phase of development. This (lack of testing) seems to be due to

- a lack of tools for e.g. Smalltalk and Java,
- the costs involved with tools,
- tools covering only limited sections of the system life cycle. In contrast EIKON however mentioned that their tool covers round trip engineering and that testing is indeed part of the tool. Perhaps developers are therefore not always aware of what is available on the market.
- a lack of user commitment (since in many of the cases the testing also needs to include acceptance testing by the user)
- the company being too small to have a testing strategy – the developers test themselves

The following testing methods were being used:

- "rigorous" testing in parallel with the old system running.
- using manual regression
- in some cases specific people are used to do testing (for example where developers do unit testing, hand over to business analysts who test and who then hand over for user testing). However in many cases it is the developers who still do the testing themselves
- implementing use cases
- code based integrated testing
- using information models as guidelines to define certain test areas.

- testing everything again, every time a change is introduced

Guidelines for testing:

- Company C, who did have a formal test design process in place, didn't think that testing was any different for OO systems. The focus is therefore on having the process in place.

- Having separate people for testing

In conclusion, it seemed that testing wouldn't have to be different for OO systems and that the strategy for testing should have been in place already if the correct software development process was in place. This leads to the issue of quality, discussed in the next section:

2.10 Quality

None of the companies interviewed felt that quality had any influence on the move towards OO. Many of the companies have their own quality process in place.

CNS and Santam both have their own standards and internal quality procedures already in place even though they are only really starting with OO now.

RMB felt that ISO 9000 certifies the process, not the software. Due to their banking environment the policy is that "if things work leave it because there are other things to do".

Outspan felt that quality is a good thing to have but in practice there is no time. (This feeling was echoed by ObjectSoft) It was however agreed that if more processes were in place it would be easier to implement OO.

There is a correlation between quality and the successful implementation of OO. (PicknPay) Testing is the beginning of it because to test there should have been requirements analysis etc. He warned against merely having quality in place for the sake of "ticking boxes".

Company B felt that implementing good OO might even improve the quality process.

At Company C they are moving towards ISO9000 not to necessarily get accredited but to be in the position to apply for accreditation. The opinion was that there was no relation between the SEI level in a company and

the success of the transition to OO – the reason being that their company was at level 0 and had implemented OO successfully. However a company at level 2 might do better but it is no prerequisite.

Momentum's representative also didn't see any correlation between quality and successful OO, specifically in South Africa, except that when using OO there are processes you go through anyway, which will improve the quality. They suggested getting a handle on the technology first and changing as little as possible at the same time

EIKON felt that it is important to use an iterative incremental development process.

In conclusion, most companies did not see the necessity to have a specific level of quality in place before moving to OO although it might help somewhat. It did become evident that as OO gets implemented correctly it may however improve the quality in the company.

2.11 Location

After an article appeared in the Computing SA stating that Cape Town is the OO capital of South Africa, it was decided to probe deeper into the issue to find out if this was indeed true.

Perhaps due to loyalty to the companies' regions, no clear decision could be made on the issue. The following were arguments given in each case:

For Cape Town:

- more development was observed to be taking place on the Intranet and Web
- people seem to have adapted to OO quicker and are more outspoken .
- the first Java certified trainer in South Africa was in Cape Town.

Against CapeTown:

- As mentioned above there is perhaps more confidence in Cape Town but companies still come to Gauteng for advice.
- There are more companies in Gauteng and therefore also more developers

- There are more jobs advertised in Gauteng in the area of OO.
- Cape Town is perhaps the OO capital in the education world but not in the real world – the proof in this case was a recent OO related conference held in both Cape Town and Johannesburg. Numbers attending made it clear that it is due to sheer volume that Gauteng is ahead in the transition.
- There seems to be a lot of new interest in OO in Cape Town. OO might be growing in Cape Town at the moment but in Gauteng it is already established.

2.12 First project

In all but one case, the first OO project was a critical project in the various companies. Reasons were the following:

- Due to the nature of the business the project was often the main project (only project)
- It takes a critical project to get management's attention and commitment

Suggestions for the first (ideal) project were:

- A project where developers can first learn everything about OO, that doesn't have impact but that seeks commitment and is of low risk.
- An "in between" project – if too small it will not matter that it worked or not, if too big, failure could lead to disaster
- A high profile project, so that people can know about it
- A short (6months - 1year) project because management will be inclined after a while to want to see results

2.13 Is OO still emerging or has the technology matured?

It is clear that although the standards are still evolving, the technology is already widely used.(EIKON) The question is now whether the technology has reached maturity or whether there will still be lots of changes. Also, what will these changes be?

According to CNS the technology is still changing rapidly and is not tried and trusted yet. There are risks. The technology was perceived to still be emerging.

According to Momentum, OO has emerged in the last year but is not mature yet – the measure to use is the variety of tools still available.

It is however exactly in this area (the tools) where most companies saw new developments in the future.

According to EIKON object technology was a thing of the early 1990s and we are already in the component-based phase.[4]

This also relates to Company C's opinion that the technology of the language is mature but that there is now growth in CORBA etc. Company C therefore sees the start of true components whereas OO up to now has not produced reusability yet.

For the future Company C sees the fine tuning of the standard bodies and a shift in development tools where you will do less coding.

According to RMB, areas where OO is still emerging includes component-based systems, distributed systems and databases. (The ODMG only published new specifications for databases recently). CORBA is also new. Therefore the next step for OO will be distributed object standards, standardised component software and pluggable items

PicknPay mentioned the San Francisco project[5] of IBM, which is considered to be where OO is heading.

According to Company B next developments will be in databases and integration between systems (information systems)

Outspan's opinion is that OO is more about defining a system. The actual systems will now move to integrated systems and artificial intelligence. What is next for OO however is more universal databases and more usable tools / systems that "ordinary" people can use.

[4] This is also reinforced by their Web page motto: "Leaders in component based technology"

[5] The San Francisco Project is IBM's move towards providing reusability: it delivers on the promise of object-oriented programming by providing a set of server-based application frameworks - it consist of about 1000 object-oriented system-independent class libraries that provides developers with the necessary building blocks for developing server-based applications.[85]

2.14 Physical area of work

No consensus could be reached on whether an open plan environment was preferable or not.

In the cases where open plan offices are used, it seemed to be a company policy rather than having relevance to development.

Arguments for an open plan environment were

- having easier access so that junior developers could learn from senior developers.

- open plan ... es interaction which goes together with the notion of having teams, ...stead of having people trying to battle by themselves.(Momentum)

Arguments against an open plan environment were

- the creation of too many disturbances. RMB mentioned the FLOW study where a researcher estimated the time for a programmer to get into "programming mode" (being productive) which is 5 minutes for good programmers and 15 minutes for bad programmers.

- instead of an open area, offices next to each other, or at least cubicles instead of an open area is proposed

However, what became clear is that the physical environment is not as important as having:

- whatever is quiet (Company A)

- people who are prepared to communicate.(Company C)

2.15 The Year 2000 crisis

In only one of the companies interviewed (Company A) did the Y2K (year 2000) problem lead to a step backwards in the transition to OO. Developers had to spend time on fixing Y2K problems instead of learning about OO. The step was in fact exactly in the opposite direction since the systems having Y2K problems were legacy systems.

In the case of Santam there were Y2K problems but these had to be solved on the side since there were fixed dates set for delivery of the new OO system.

All the other companies were already Y2K compliant and therefore had no problems.

EIKON commented that the Y2K issue is real proof that procedural methods did not work. On the other hand, the fact that companies must now often take a step back from moving to OO, can influence their (EIKON's) business because new development is put on hold in these companies.

2.16 OO insertion: Bottom up or top down?

Arguments for an insertion from the developers' side were:

- Sometimes developers know more and have more experience.(CNS)
- people at the bottom will find fault if they do not like the language or methodology and will not use it. (an interesting comment from RMB: if with Smalltalk there are problems it's the language, if it is a C++ project, it is a management-related problem). This was also the experience at Company C.

Arguments for an insertion from management's side:

- the need for an appropriate champion to push the move towards OO (RMB)
- a strategic decision is necessary to avoid having islands of development, therefore you need to insert from the top down because a large financial investment is required at first.
- fighting management is not easy

The majority's opinion was that the insertion needs to come from both sides. In contrast, in one case it was also proposed that it should be from middle management (or project leaders) outwards. It therefore seems to be a company-specific decision.

2.17 A difference in management

There was no consensus whether the management of OO products is indeed different from the management of other projects. The opinions expressed also did not reflect a clear separation between the experienced and less experienced companies.

The following arguments were given to support the change in management:

- With OO there is a short turnaround (from bug report to fixing to new release) whereas with structured methods there are long processes. This changes the planning. (RMB)

- The difference in management style is related to library reuse, testing etc.(Company A)

- It is more about collaboration.

- According to EIKON the management is so different that they even have a course on this change.

- The difference in management stems from the paradigm shift, having an iterative approach and doing small "builds" which makes it difficult to manage.(Momentum)

- COBOL programmers have a different attitude from the new OO programmers who are "go getters", therefore there are now different types of people to manage which makes it difficult.(PicknPay)

Arguments against a change included:

- OO is just a tool (Company B)

- It is more in the implementation where there is a difference (ObjectSoft)

2.18 Evolution or Revolution?

Most companies agreed that the revolutionary approach was preferable to the evolutionary approach for the following reasons:

- RMB mentioned the problem where it is easy for developers to use C++ or Smalltalk procedurally. The only way to solve this problem is by getting new developers and using the old ones for maintenance (the COBOL people)

- EIKON mentioned that the waterfall approach does not work for OO and therefore the new process needs to be adopted together all at once

Arguments against the above came from Momentum's representatives saying that OO is the pinnacle of maturity to which all the other methodologies have built up to.

2.19 Training

2.19.1 Universities and technikons

The opinions seemed to be unanimous:

- The universities are not providing students with adequate OO training: training is not practical enough, should be more extensive (some universities in other countries even offer the opportunity to "majo. in OO).
- There was a suggestion that perhaps people from the industry should also be used for lecturing. Courses are too theoretical. Often graduates still need to be sent on courses when joining the company.
- In contrast the technikons do not offer a basis for understanding the underlying theory.
- The good people seem to be in the industry - the issue is salary related.

2.19.2 Graduates or non graduates

- None of the company had any bias about employing only graduates or non-graduates. CNS mentioned that graduates are often productive quicker but also tend to leave the company earlier because they are marketable and can find jobs easier than non-graduates can.
- At Company A, they prefer to use graduates for long term appointments and non-graduates for short assignments.
- An interesting remark was guarding against employing "too clever" people – the reason being that development done in a too clever way can lead to maintenance problems.
- There is always the prejudice that graduates learn new things easier and can work independently but perhaps this is expected and is therefore like a self-fulfilling prophecy.(Company B)

2.19.3 Requirements

- The requirement is therefore not for people to be graduates or not graduates, it is rather a case of IT people also having business skills and vice versa.

- People with practical experience are in demand.
- People should fit into the culture – that will determine whether they will stay
- attitude is more important than aptitude (PicknPay)
- at Company C there is no prejudice about whether people have previous experience in C or not – they believe that if they are good in C they will also probably be good in C++[6].
- in contrast PicknPay mentioned that the COBOL people will have too many learning curves and for OO one should rather use new people. (echoed by RMB)

2.19.4 Training companies

Outside training companies are used in most cases. In two cases the companies use inside training for big teams and outside training for smaller teams. Arguments in favour of outside courses were: getting developers out of the building to prevent them from worrying about work that need to be done and concentrate on the course.

In two cases the companies rely heavily on internal mentoring, where new people are assigned to older people for guidance.

Most companies were happy with the standard of the training companies. However the following problems prevail:

- CNS complained that courses had prerequisites, were too long, the trainers made assumptions on what people knew about Windows etc, while the developers were actually mainframe experienced.

[6] This is reinforced by the following comment: "The most important factor in getting OO technology inserted was skill leverage....the good procedural designers became the good object designers, probably because they had abstract reasoning capability" [56].

- There is a lack of confidence in South African consultants and overseas consultants are used.
- Courses are too limited in South Africa but the quality of the courses are good.
- Quite a few companies thought that the training companies often have less experience than they do and therefore they will rather use individuals that are good.
- RMB also thought that the training organisations in South Africa are not sophisticated enough. For example there is no sufficient training for CORBA and the ones who do train are partisan and force their clients to use their product.

2.19.5 Train management?

The guideline here is that management should at least be able to speak the same language and understand the technical issues for budgeting purposes. RMB's representative however warned against training management, arguing that a little knowledge could be dangerous and that there are situations where the architects should be trusted with decisions.

2.20 The Global situation

There were mixed opinions about the situation in South Africa in comparison to the rect of the world.

Many people believed that South African companies are not that far, if at all, behind the rest of the world:

- At various interviews it was mentioned that the demand for South African developers overseas suggests that in general we have very good technical capabilities. Developers here are experienced in a wider range of skills. (although this may change due to the current "skills drain" out of the country)
- OO progress in the financial sector is faster in South Africa due to our good banking systems.

- Overseas there is more data, which always makes it difficult to move over to new technologies.
- There are islands of excellence here that are just as good as overseas - due to numbers there are just fewer islands here.
- the Internet makes it easy now to stay with the overseas companies.

The following suggested that the good capabilities we have here are not applicable for OO development:

- It was felt that there is a lack of awareness of OO here. The number of people attending conferences here compared to other countries confirms that.
- Compared to people like Fowler, Booch, etc. there is no OO guru in South Africa.
- There is no support for e.g. Smalltalk here.
- It is also in attitude that we are behind
- OO hasn't been implemented as widely here because there was never an urgency to do so.
- South Africa is not where "it is happening"
- in product utilisation we are on par but in new development we are behind.(Outspan)
- Overseas there are various institutions where OO is already a mature technology having large OO systems in place.
- Lastly, EIKON provided the following evidence: generally South Africa is 18 months to 2 years behind the US in various technologies. In the US, in 1994, 21% of companies had an OO strategy, in 1995 45% and in 1996 94%. The opinion is that we are now at the 1995 mark.

In conclusion, it is this last comment that suggests that even though as a whole South Africa has excellent technical capabilities, we are still behind in the OO arena. It was also mentioned that this situation might get worse, while young people, who are keen to learn but who are also the most mobile, are leaving the country at an alarming rate.

2.21 Companies providing an OO service

In general the companies interviewed did not made use of the services of companies providing consulting in the transition to OO (except where using individuals) Reasons were the following:

- The companies themselves are already above the level of service offered by these companies and have more experience themselves.
- If any, the smaller companies are used but not the big companies since these were typically traditionally COBOL-experienced who have since seen that OO makes money and therefore want to jump in without offering experience.
- Many of the companies interviewed are not aware of these companies, often because they just have not been looking (Company A) (Company B)
- There are high costs involved
- These companies often only focus on certain market sectors.
- South Africa is driven too much by suppliers who think they are consultants
- Companies often make use of individuals only
- It is felt that these consulting companies would not understand the culture of the company using their services.

One company that was repeatedly mentioned (in a good light) was Borland's Real Systems division. However, again it is a case of specific products being supported. The need for product-independent services became evident. Everybody thought that there was definitely a market for this type of service.

2.22 Databases

Everybody agreed with the finding in the Cutter Consortium report stating that smaller companies are moving towards OO databases faster than large companies. The motivations were:

- where there are less people, they take more pride in what they do because they own it

- in large companies there are more rules (more bureaucracy).
- Due to politics and the lack of communication, management in large companies are often uninformed and are influenced by the vendor's propaganda.
- In large organisations there are myths and prejudice about OO databases e.g. that these databases cannot handle large volumes of data
- small databases are easier to convert, whereas with large ones people are reluctant to change if there is no need.
- It's easier and cheaper to change in smaller companies due to less data
- large companies cannot afford down time
- large companies are restricted by the legacy systems.
- small companies are less conservative

Regardless of company size, the following situations might however suppress the move to OO databases:

- In some cases, companies often have a few or one client(s) and are forced to use what the client demands, which might not necessarily be an OO database.
- There is a feeling that OO databases have not featured in business environments yet – it rather presented more problems. Object oriented databases can still not compete with relational databases in business environments

2.23 Company profile

2.23.1 Company size

The opinion was that smaller companies would generally find it easier moving to OO for the following reasons:

- Large companies typically have many products in other technologies established in the market
- Large companies sit with old (legacy) systems to maintain
- Large companies needs more co-ordination in the effort and communication is better in the small companies
- Large companies have more people who are resisting the move whereas small companies have fewer people to get commitment from.
- Small companies need a radical approach to be successful and attract attention which is what OO gives them
- In large companies a concept such as OO will not be pervasive and will not be accepted throughout (banks are conservative). In a smaller company, if a technology is accepted, it is done throughout
- Small companies have fewer mistakes to manage.

2.23.2 Project size

In general it was felt that smaller project team sizes would do better in the transition towards OO.

Arguments in favour of small projects were:

- as with company size, if there is a change there will be less to change
- better communication exist in small projects - there is no need for formal communication as is required for large projects
- a group needs to be cohesive. Everyone has their own opinions and different experience to share which in a large group can make the process too long.
- In one specific case it was mentioned that before, using Adabas, although there was a large project involved, the team could be broken up easier whereas now, using Visual Basic, the system development is more integrated and therefore difficult to manage.

Arguments in favour of large projects:

- costs can be handled easier
- OO comes to its true value better
- OO is better at handling large problems and it scales well

2.23.3 Project lifetime

No consensus was reached.

In favour of short projects when developing OO systems

- if the project takes longer than 9 months there is a problem

In favour of long projects:

- nothing sufficiel can be completed in a year. Lots of companies think they are implementing OO, but are actually implementing Rapid Application Development using Delphi etc. It is only when the projects get bigger that companies realise that they cannot simply hack a solution together
- by default one needs more time and need to design ahead which is why longer projects are doing better.
- smaller projects don't need to be as formalised - it only needs to work whereas long projects deal with more complexity which is where the real test comes in.

2.24 Market sector

In general people agreed with the finding that it is the financial, IT and retail sectors that are ahead in the adoption of OO at the moment. Reasons given were:

- financial systems are made for OO because the software is complex. OO provides a way to handle this.
- In financial systems there is lots of pressure. New releases are constantly required. OO allows for making these changes.

- South African banking seems to be advanced compared to the rest of the world and therefore these systems will also be using the advanced technologies.

- the IT, retail and financial companies often work with one big system rather than numerous small ones – therefore there is more of a focus on the technology used

- the dynamic environment found in IT financial and retail companies, implies that there is no luxury of spending years on a project. There are also new requirements all the time in these sectors

- an interesting observation by EIKON was that although the notion of financial, IT and retail being advanced in OO is true, it is also true for many technologies because these kind of companies need a competitive advantage.

- The financial companies have lots of money & need good information systems

Arguments against the finding were:

- The distinction should rather be made between technology driven (e.g. insurance companies where there is competition) versus non technology driven companies (e.g. monopolies) (Outspan)

- The finding about retail being advanced was questioned by Company C, arguing that perhaps these companies make use of a lot of outsourcing which means that the focus again falls on the IT sector rather than retail as such

- Surprisingly it was one of the financial companies that didn't agree, arguing that more resistance exists in the financial sector whereas developers constantly experiment with new technologies in the more technical sectors (military, manufacturing, etc).

2.25 Methodologies

Where any methodology was used, UML was the most popular methodology being used, although most of these companies created their own customised version of it. In many companies no methodology was used, highlighting situations where the move was made to the new tools on the market and not really to the OO methodology.

The following observations were also made:

- In most cases the methodology is used for the whole development life cycle, but it seems to be more successful from software analysis to testing

- Many companies mentioned that the more involved one becomes in OO the more one realises the importance of using a methodology.

- Although UML has become the standard now, Company B mentioned that in terms of maturity at least the Shlaer Mellor methodology was very formal and you were lead through each phase, whereas UML does not always describe the process, rather the notation. This opinion was also echoed by Momentum, saying that design is not just about notation, but also about the process.

- No specific process will give all the answers – the feeling was expressed repeatedly that "not all the ticks on the checklists always apply" i.e. use what applies to you.

2.26 CASE tools

With the exception of EIKON, the companies questioned the state of the CASE tools available at present. Most of these companies therefore do not use any CASE tools and if they use any, it is mostly for the documentation and capturing of designs done on a white board. The following complaints were given:

- CASE tools often help with the first round and then when the design changes the whole process has to be redone.

- CASE tools work fine for 80% of what you want to do and for the last part you are left on your own – this is also probably the most important part of the design.

- The tools are very expensive, aimed at the big users and companies cannot justify purchasing it. For a drawing tool it is very expensive - companies would not object to paying a large amount if the tool supported the real engineering process.

- The design process takes longer when using CASE. There is also no time to experiment and get to know the tool. Tools could give you the wrong results if you do not use it correctly.

- CASE tools are not flexible enough
- In the cases where a good CASE tool seemed to be at hand, management could not be convinced to spend the money on something that they felt "they cannot see".
- When you've mastered the process - that's when the CASE tool will help but that is as a result of having the process, not the CASE tool.

Where present, the tools were not used for code generation. New areas investigated were using tools for deployment

2.27 Metrics

Metrics seemed to be a less important issue on the minds of most the people at companies interviewed. Very few companies used any form of metrics.

The feeling was that the metrics available do not measure up to the expectations. Complaints were:

- There are very few metrics available for OO
- What is available for OO is counterproductive
- It is not easily understandable and usable
- Attention paid to metrics should be related more to the business needs rather than specifically to OO: speed, flexibility and whether it is what the client wants is still more important
- There is no consistent standard for the usage of the metrics available even if the tools are good. Therefore no comparison can be done with other projects or companies. (This seems to be true for all disciplines since at Safmarine the comment was again made that there is no standard for measuring maintenance and development productivity and that there is also no industry standard for using function points[7]. This problem therefore seems to be universal and not only OO related)

[7] First used in 1979, function points provides an objective measure and is used in the planning of software development. It is derived in a number of stages and the final result is a single number or index which measures the size and complexity of the software developed.

- management often does not see the need to pay for something that is not tangible.

- In the case of Company C they used no CASE tools because there was no money available, but felt that there were good tools available.

- In Momentum's case the company is at level 1, using hero programmers, and it is felt that metrics goes alongside with having quality in place

Guidelines for using metrics are the following:

- metrics are important, especially where sceptical management is involved

- one person should be dedicated to metrics since it is complex

- there are good tools available but you should look at your own situation individually and not take the tool's numbers at face value e.g. if the tool says a score of 10 is bad you might get 15 and it might be good for your situation

2.28 Organisational structure

In half the cases studied the organisational structure has not changed at all. Most people however thought that such a change was necessary and would happen in the future.

The opinions at EIKON and PicknPay were that there are different roles involved in OO as compared to procedural development, and that you now need a system architect who can also act as mentor, as well as a project manager, developers and a quality assurance team.

In the cases where the structure changed the following were the reasons:

- In one case the structure came into being (there was no structure before) from putting the processes in place. It was therefore not directly linked to the move to OO.

- In another case a general company restructure took place (to a team oriented approach) simultaneously with the move to OO and is therefore not really due to the transition to OO.

- In the case of RMB the change was done to a flat structure where developers work on more than one type of project. Again it seemed that this change did not relate to the change to OO.

It is therefore not clear how important the organisational change mentioned in various articles[37][43][47] really are in South African companies.

2.29 Speed of adoption

The following factors were mentioned to be the factors influencing the speed of adoption of OO in companies:

- Age of the developers: young people want to learn, older people are indoctrinated with structured principles and have prejudice. Together with this goes arguments for people fresh from university rather than people with years of COBOL experience.
- Knowledge of the process and the business
- Commitment and willingness ("buy in" from the people) – this was mentioned many times
- In contrast to the above argument it was mentioned that it is often the old people who have experienced the problems with the old technologies that are now willing to learn the new.
- In some companies unfortunately the "critical mass factor" is important – the company will not use a technology if the rest of the market does not.
- In one case it was felt that it is more a case of putting the process in place that made the difference rather than OO related issues.
- Technical skills – surprisingly this seemed to be last on everybody's list

2.30 Language

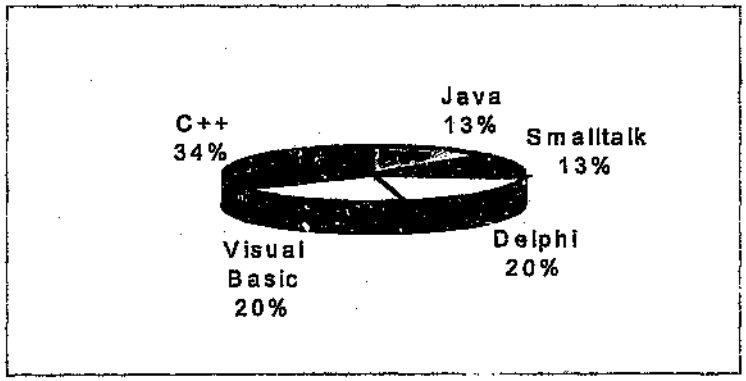


Figure 3 Language Usage

The languages used in each of the companies are given in Figure 3.

Reasons for the specific language chosen were:

Java:	platform independence and pure OO
Smalltalk:	pure and dynamic language, garbage collection
C++:	moved there from C, Delphi was not available at the time the process started.
Delphi:	in the case of Company C it is used for prototyping while C++ is used for the development.

It seemed as if the companies using Java, Smalltalk and C++ had more experience in OO while companies using for example Visual Basic were using the tools. Perhaps this stems from the kind of development where