South African Computer Journal
Number 23
July 1999

Suid-Afrikaanse Rekenaar-tydskrif
Nommer 23
Julie 1999

Computer Science and Information Systems

Rekenaarwetenskap en Inligtingstelsels

The South African Computer Journal
An official publication of the Computer Society of South Africa and the South African Institute of Computer Scientists

Die Suid-Afrikaanse Rekenaartydskrif
'n Amptelike publikasie van die Rekenaarvereniging van Suid-Afrika en die Suid-Afrikaanse Instituut vir Rekenaarwetenskaplikes

World-Wide Web: `http://www.cs.up.ac.za/sacj/`

Subscriptions

	Annual	Single copy
Southern Africa	R80.00	R40.00
Elsewhere	US$40.00	US$20.00

An additional US$15 per year is charged for airmail outside Southern Africa

to be sent to:

Computer Society of South Africa
Box 1714, Halfway House, 1685
Phone: +27 (11) 315-1319 Fax: +27 (11) 315-2276

Guest Editorial
Computer Science and Information Systems: The Future?

Philip Machanick

Department of Computer Science, University of the Witwatersrand, South Africa
philip@cs.wits.ac.za

1 Introduction

As president of the South African Institute for Computer Scientists and Information Technologists (SAICSIT), I have visited a number of campuses and companies, in an attempt at arriving at a general assessment of the state of our subjects in South Africa.

An issue which I consistently pick up is that while everyone seems to think that computer-related skills are extremely important and in short supply, our academic departments are also extremely under-resourced.

At the last Southern African Computer Lecturers Association (SACLA) conference (28-29 June, Golden Gate), I had the opportunity to discuss the problems other academics see. This editorial lists some of the problems reported at SACLA, and proposes a way forward.

2 Problems

At SACLA, I led a discussion of problems seen in our academic departments.

There was wide agreement that both Computer Science (CS) and Information Systems (IS) departments were under pressure to increase student numbers (massification), and were seen as cash cows to prop up less popular subjects. It was broadly agreed that staffing was a critical issue: too few posts for the workload, salaries way out of line with industry (half or less, as compared to the US, where an academic salary may be 80% of an industry salary). Recent graduates often make more than professors which makes it hard to persuade our students to become academics (even to do higher degrees). Attracting a recent PhD with a sense of adventure is may be possible, but attracting experienced people used to earning a salary in a strong currency is hard. IS jobs are worse than CS, as the skills required are more like those in business. Support staff salaries are an even harder issue: their skills relate even more directly to job descriptions in industry.

A problem in addressing our concerns is that we are so overworked that we don't have time for "politics": academics with no students have time on their hands, but we don't. More industry support not only with directly addressing problems but with taking on university administrations would be useful, but they too have major problems and don't have free time.

3 Solutions?

Solutions are harder to identify than problems.

The SACLA session ended with a proposal that we conduct surveys of our institutions and businesses, to find out what the problems are, as a starting point for going to university administrations, government and business.

Another idea was to attempt to find common cause with business in taking on problems they have in common with academia, including the skills shortage, the insufficient capacity of our education system, and dealing with employment equity.

One of our biggest difficulties is to free up time to deal with issues such as resource allocation within our universities. The "competition" is frequently other academics with time on their hands, since they have too few students, and therefore are in a position to spend time looking after their interests.

What is needed now is some thought about how to pull ourselves out of the mess we are in. In particular, we need strategies to exploit our strengths: our high demand among students, the high demand for the skills we produce and the ubiquitous applicability of computer technology.

Given the wide use of computers, it would seem obvious that our areas should be strongly supported by a range of role players, yet the fact that so many different groups are interested in computer technology in one way or another has tended to fragment efforts to enhance our industry and academic institutions.

Clearly, from conversations I have held, some departments are in much better shape than others. Even so, some kind of collective effort is likely to achieve more results than if we allow ourselves to be pushed around as individuals. Addressing the fragmentation of efforts seems a worthy goal in itself, to reduce duplication and contradictory goals.

I appeal to anyone who has constructive ideas on how to take our subjects forward to contact me. Let us work on building ourselves up. The economy depends on us, much more than on most other academic disciplines. It's time we made that point, and made it strongly.

SAICSIT'99

South African Institute of Computer Scientists and Information Technologists

Annual Research Conference 17-19 November 1999

Prepare for the New Millennium

Is there life after y2k?

Mount Amanzi Lodge, Hartebeespoort

near Johannesburg and Pretoria

keynote speaker: Barbara Simons, ACM President

and many other local and international speakers (academic and industry)

Call for Participation

papers in a many areas of Computer Science and Information Systems are expected

Price Waterhouse Coopers prizes: Best Paper R10000 • Best Student Paper R5000

please check the conference web site for accepted papers:

http://www.cs.wits.ac.za/~philip/SAICSIT/SAICSIT-99/

To Register

go to the conference registration web page:

http://www.cs.wits.ac.za/~philip/SAICSIT/SAICSIT-99/reservation.html

or contact **SAICSIT'99 Secretary** *for details:*

Department of Computer Science, Senate House 1137
University of the Witwatersrand
Jorissen Street
Wits, 2050
South Africa

phone (011)716-3309 fax 339-3513 (international: replace 011 by 27-11)

saicsit99-info@cs.wits.ac.za

Dates and Publication Details

early booking deadline: 14 September • on-site registration starts: 17 November 1999

workshops, tutorials 17 November 1999 • paper sessions: 18-19 November 1999

papers will appear in a special issue of South African Computer Journal

sponsors

NRF PRICEWATERHOUSECOOPERS

Progressing towards Object Orientation in South Africa

M. Jansen van Rensburg

Software Engineering Applications Laboratory, Electrical Engineering, University of the Witwatersrand, South Africa,
mjvanren@tsamail.trsa.ac.za

Abstract

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: *D.2*

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.

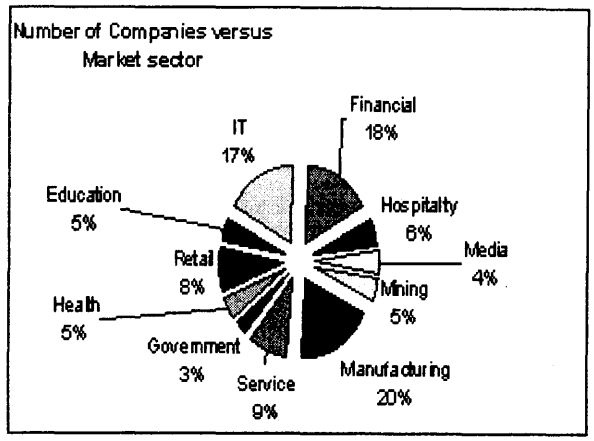

Figure 1: Companies versus market sector

3 The Telephonic Interview

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

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

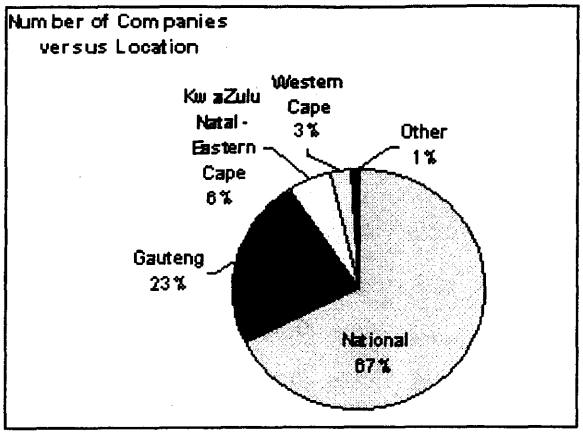

Figure 2: Companies versus location

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.

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 incremen-

tal 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 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

[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 [5]

[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].

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 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 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

[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 [29].

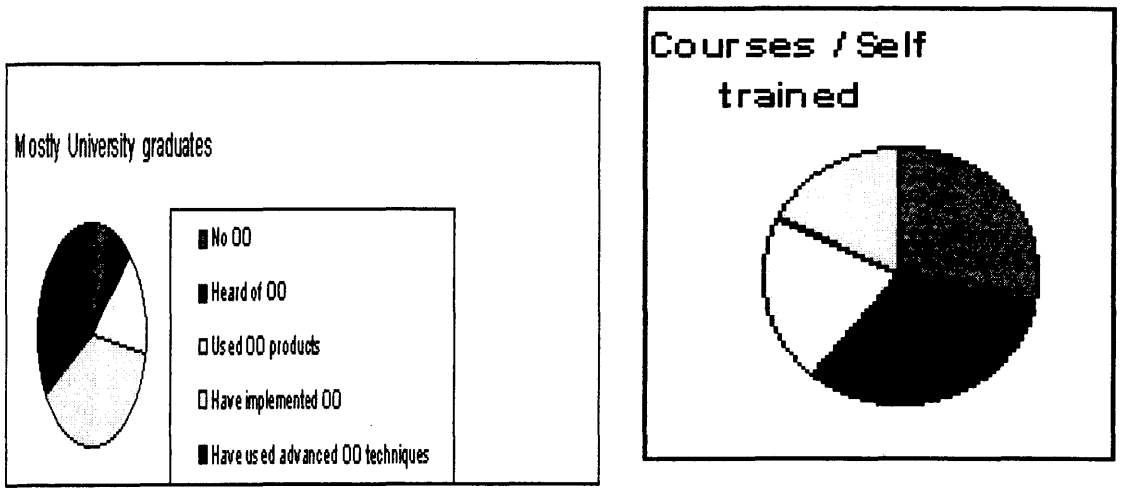

Figure 3: Training and OO

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, Vayda [32] 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. 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".

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

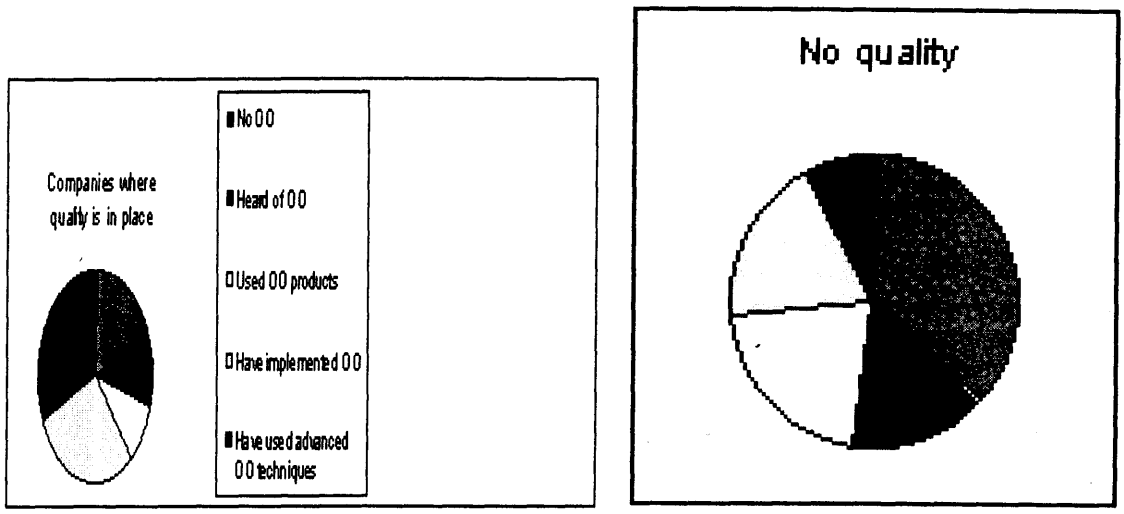

Figure 4: OO and Quality

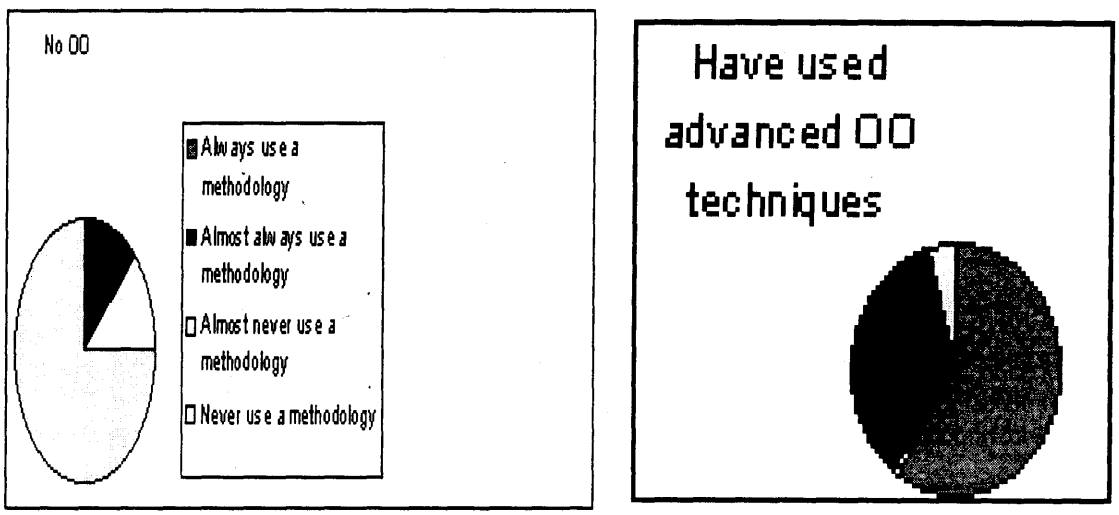

Figure 5: OO and methodologies

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:

- 11Mistakes 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.

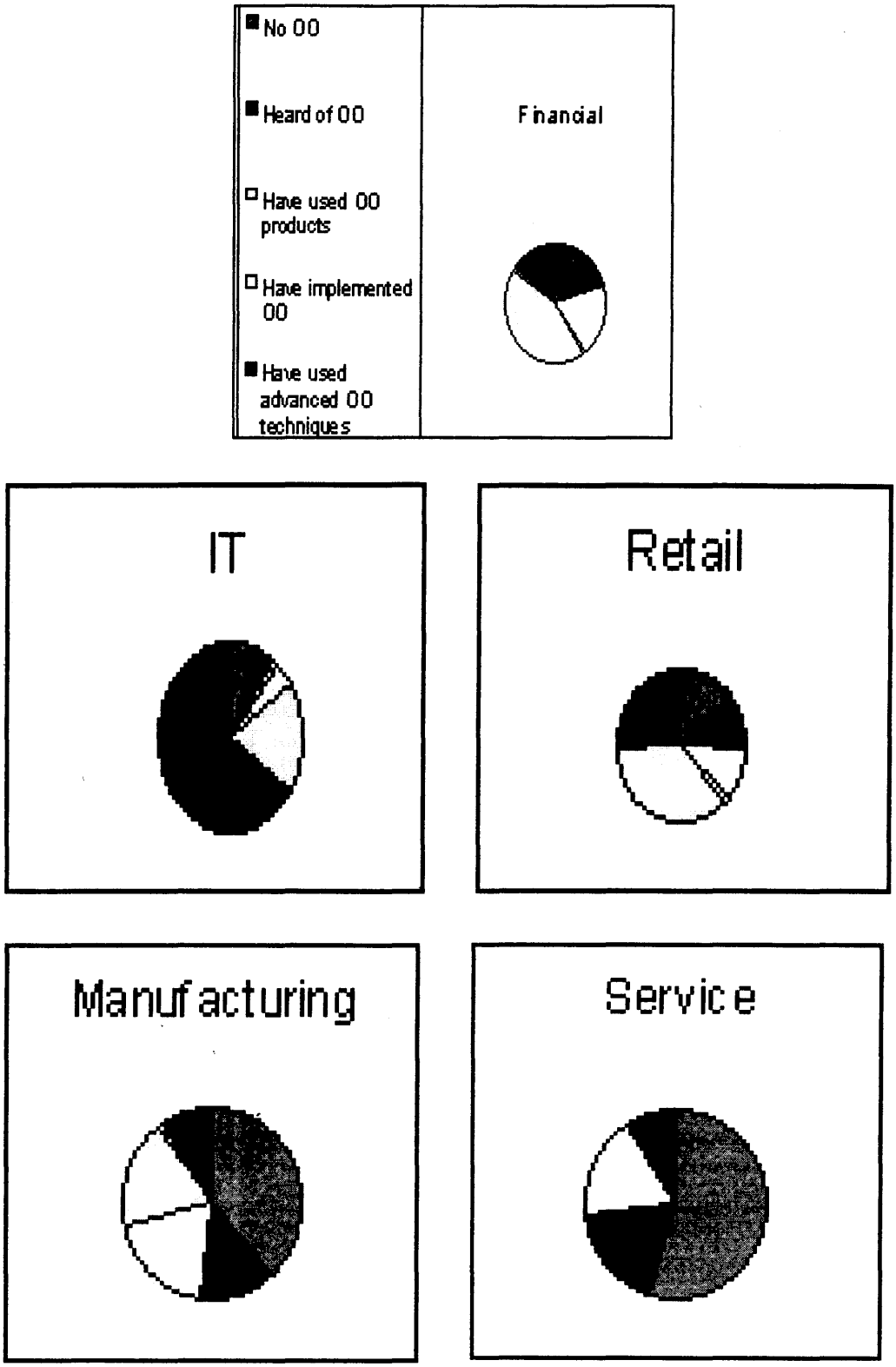

Figure 6: OO and market sector

7.1 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 [12]. 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 navigate 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 I 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

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

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.
- 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

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

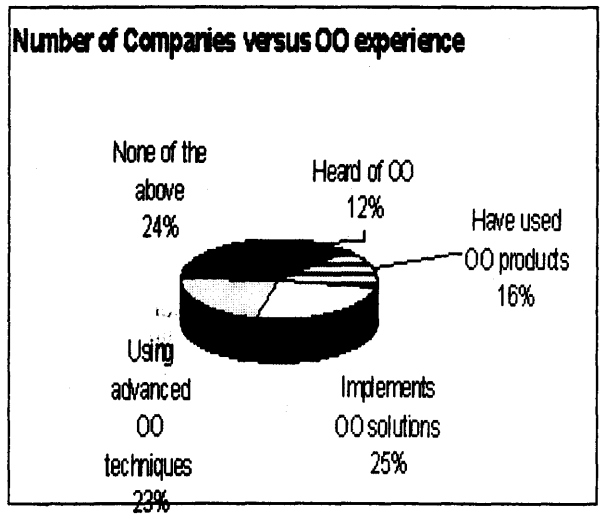

Figure 7: OO experience in South Africa

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 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.

In other countries there are various institutions where OO is already a mature technology and large OO systems

[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].

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

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.

References

[1] T.R. Arnold and W.A. Fuson. 'Testing "In a Perfect World"', *Communications of the ACM*, **37**(9):78-86, (1994).

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

[3] V. Basili, L.C. Briand and W.L. Melo. 'How reuse influences productivity in object-oriented systems'. *Communications of the ACM*, **39**(10):104-116, (1996).

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

[5] E. Casais. 'Re-Engineering Object-Oriented Legacy Systems'. Journal of Object-Oriented Programming, January:45-52, (1998).

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

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

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

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

[10] I. Graham. 'In Search of the Three Best Books'. Journal of Object Oriented Programming, September:43-45, (1997).

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

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

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

[14] T. Korson. 'Managing the Transition to Object-Oriented Technology'. In OOPSLA'91 Proceedings, 55-62, (1991).

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

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

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

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

[19] D.R. Moreau and W.D. Dominick. 'Object-Oriented Graphical Information Systems: Research Plan and Evaluation Metrics'. The Journal of Systems and Software, **10***:23-28, (1998).*

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

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

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

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

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

[25] C.M. Pancake. 'The Promise and the Cost of Object Technology: A five-year forecast'. *Communications of the ACM*, **38**(10):33-49, (1995).

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

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

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

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

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

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

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

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

Received: 8/1/99, Accepted: 4/3/99

Notes for Contributors

The prime purpose of the journal is to publish original research papers in the fields of Computer Science and Information Systems, as well as shorter technical research notes. However, non-refereed review and exploratory articles of interest to the journal's readers will be considered for publication under sections marked as Communications of Viewpoints. While English is the preferred language of the journal, papers in Afrikaans will also be accepted. Typed manuscripts for review should be submitted in triplicate to the editor.

Form of Manuscript
Manuscripts for *review* should be prepared according to the following guidelines:

- Use wide margins and $1\frac{1}{2}$ or double spacing.
- The first page should include:
- the title (as brief as possible)
- the author's initials and surname
- the author's affiliation and address
- an abstract of less than 200 words
- an appropriate keyword list
- a list of relevant Computing Review Categories
- Tables and figures should be numbered and titled.
- References should be listed at the end of the text in alphabetic order of the (first) author's surname, and should be cited in the text according to the Harvard. References should also be according to the Harvard method.

Manuscripts accepted for publication should comply with guidelines as set out on the SACJ web page,

`http://www.cs.up.ac.za/sacj`

which gives a number of examples.

SACJ is produced using the LaTeX document preparation system, in particular LaTeX 2_ε. Previous versions were produced using a style file for a much older version of LaTeX, which is no longer supported. Please see the web site for further information on how to produce manuscripts which have been accepted for publication.

Authors of accepted publications will be required to sign a copyright transfer form.

Charges
Charges per final page will be levied on papers accepted for publication. They will be scaled to reflect typesetting, reproduction and other costs. Currently, the minimum rate is R30.00 per final page for contributions which require no further attention. The maximum is R120.00, prices inclusive of VAT.

These charges may be waived upon request of the author and the discretion of the editor.

Proofs
Proofs of accepted papers may be sent to the author to ensure that typesetting is correct, and not for addition of new material or major amendments to the text. Corrected proofs should be returned to the production editor within three days.

Letters and Communications
Letters to the editor are welcomed. They should be signed, and should be limited to about 500 words. Announcements and communications of interest to the readership will be considered for publication in a separate section of the journal. Communications may also reflect minor research contributions. However, such communications will not be refereed and will not be deemed as fully-fledged publications for state subsidy purposes.

Book Reviews
Contributions in this regard will be welcomed. Views and opinions expressed in such reviews should, however, be regarded as those of the reviewer alone.

Advertisement
Placement of advertisements at R1000.00 per full page per issue and R500.00 per half page per issue will be considered. These charges exclude specialised production costs, which will be borne by the advertiser. Enquiries should be directed to the editor.

South African Computer Journal

Number 23, July 1999
ISSN 1015-7999

Suid-Afrikaanse Rekenaar-tydskrif

Nommer 23, Julie 1999
ISSN 1015-7999

Contents