

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/

more knowledge is necessary of the design principles itself and merely using a tool is not good enough.

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

2.31 Mistakes

The following are the problems that companies ran into along the way. Many of the mistakes may seem not to be directly OO related, but lead to a delay in the move to OO.

- Lack of proper design and not using a methodology
- The importance of the methodology was not stressed sufficiently
- Lack of skills
- using a relational database – in one case 30% of the code and 60% of the performance were compromised just to handle the conversion.(RMB)
- Not having the luxury of time to go into a smaller project first
- Due to movements and demands in the market, not having the luxury of time to investigate different tools first (this could be significantly different in companies overseas)
- Many companies are still operating at a SEI level of one, thereby having hero programmers – if these programmers leave there is no one to take over. Although this is not directly OO related, it does influence the speed at which an adoption of OO can happen.
- Lack of proper documentation
- Lack of full time project management
- Being too calendar driven: this was mentioned numerous times – dead lines had to be met regardless of the quality of the system being developed.
- Lack of communication between the users and developers

- Having no time for training
- Using rapid application development tools without having a proper design

The following were the typical problems experienced:

- Having to deal with a new technology and a new way of thinking
- A new terminology to deal with, especially for traditional mainframe developers
- Speed requirements that have to be met, which is why the hardware chosen is just as important.
- At Company B it was found that although they used OO for handling complexity, the system analysis was still functional in nature. It was difficult to find a transition from the one to the other. There are 3 orthogonal ways of looking at the life cycle: in a functional way, a datacentric way and in an object oriented way. There is no automatic process for integration of the whole life cycle.

2.32 Guidelines

The following guidelines were given to companies contemplating the move to OO:

- Do not just jump in – do proper research since there are many products on the market that might not be applicable
- The three tier design[8] is important and should be done correctly

[8] The three-tier client-server architecture is replacing the classic two-tier architecture used in the past.. In the two-tier design, the client always handled data presentation and the server managed the database. If the business logic is then implemented on the client the software becomes platform dependent, if it's on the server the server gets overloaded. The three-tier design provides an additional separation of the business logic from the database and the actual presentation, so that by using CORBA the business objects can be put wherever the resources are available for support.

- look at real solutions - numerous companies offer consulting services, but making the wrong choice can lead to huge losses.
- OO should be a strategic decision from the top, with formal training and for the whole project.
- Budget for enough time. Management should be made aware that development will take longer and that the company will only see the benefits later.
- One should also make the transition to OO for the right reasons and not for choosing the "flavour of the month" technology
- stay abreast with the new technologies
- OO cannot work without senior management commitment.
- OO does not work without a process.
- a methodology needs to be present else OO becomes hacking in a different form
- use mentoring – you need a guru and need to understand design patterns
- regarding patterns: they will not solve everything – it captures the essence of a good solution. One needs to use it in conjunction with common sense – do not attempt fitting a pattern somewhere just to fit it
- one must partner with a successful vendor,
- it is the people, not the technology that is important: one must use high flyers as developers (young people who are open minded)
- management should be in touch with the new technology, else developers will try and fool them

2.33 A comparison with Safmarine

Lastly, the situation at Safmarine will now be analysed in detail to see what there is to learn from the transitions made to other methodologies:

Success factors:

1. Firstly, the comment was made that it is the people involved that make the difference, not the tool.

2. Although not the most important, the tool does play a role in the success of the transition as the following shows:

   - The tool used in this case provided support throughout the whole system development life cycle, from strategic planning to maintenance
   - The tool provides a formal method but can be customised to allow the user to choose the steps necessary
   - Metrics were provided for measuring of productivity
   - Configuration management is part of the tool.
   - Code generation was possible in the language of their choice
   - Reuse was measurable and was found to be 25%

3. There seemed to be a well-defined development process in place:

4. Regarding technology insertion, when the final selection was made, they had technical people but also management involvement when doing research about which technology to use – the final decision was taken by vote.

The mistake they made: targeting a date to go live regardless of the quality of the work done. The comment was made that users do not remember a delay in delivering but they do remember a smooth release. This mistake was also mentioned in the case of Company C - it seems to be universal in nature.

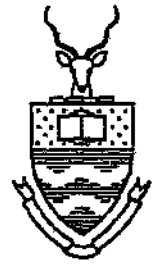
Guidelines from Safmarine included:

- use dedicated users for testing

- full commitment from top management is vital
- people make it happen not the tool

It is clear that each of these guidelines as well as the lessons learnt can also be applied directly to the transition to OO.

■



NDM QMS

NDM - The Overseas Questionnaire

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 Introduction 2

3 Overseas questionnaire 3

3.1 The company 3

3.2 Where we are now 3

3.3 Why OO? 4

3.4 Company profile 5

3.5 Soft Issues 6

3.6 First project 7

3.7 Training 7

3.8 Reuse 8

3.9 Language 9
3.10 Legacy 9
3.11 Testing 9
3.12 Quality 10
3.13 Methodologies 10
3.14 CASE 10
3.15 Metrics 11
3.16 Timing 11
3.17 What is next 12
3.18 Problems 12
3.19 Guidelines 12

Change History

Configuration Control

Project:	NDM QMS
Title:	NDM The overseas questionnaire
Doc. Reference:	C:\MSC\TP\NDM311.100
Created by:	M Jansen Van Rensburg
Creation Date:	8 July 1998

Document History

Version	Date	Status	Who	Saved as:
0.01	98\05\06	Draft	MVR	NDM311.010
1.00	98\09\23	Approved	MVR	NDM311.100

Revision History

Version	Date	Changes
0.01	98\05\06	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 a record of the questionnaire used during interviews held with companies in the United States during September 1998. The interviews were held with the purpose of comparing the progress that has been made in South Africa and in the USA in the transition to Object Orientation.

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 USA 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 Overseas questionnaire

Company Name:
Contact Name:
Company Detail (telephone/address):
Date:

In each issue of discussion I would like to get information not only about your specific company but also about what the general trends are, that exist.

3.1 The company

The company size:

Can you briefly describe the activities in the IT department?

3.2 Where we are now

What is your opinion?

According to Pickering's survey[79], the situation in 1994 could be described as follows:

Table 1

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

How does this compare with the current situation in your opinion?

3.3 Why OO?

What were the reasons for moving towards OO in your case?

South African companies' reasons were:

- *"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.[76]

3.4 Company profile

3.4.1 Market Sector

Would you say that certain market sectors do better than others in the transition to OO?

(In South Africa it seems as if the financial, retail and IT sectors are more advanced.)

3.4.2 Company size

Do you think that certain company sizes are more suitable when moving towards OO?

(In South Africa the feeling was that small companies would do better, needing a radical approach to be successful and attract attention which is what OO provides. Large companies have bureaucracy and more people to convince.)

3.4.3 Project size

In your experience, is there a relationship between the project size (number of people) and the success in the transition towards OO?

(In South Africa, as with company size, it seemed as though small projects were preferable, having fewer different opinions, better communication and less to change.)

3.4.4 Project Life time

In your experience, is there a relationship between the project lifetime (number of months) and the success in the transition towards OO?

(No significant conclusions could be drawn in South Africa. My feeling is that lots of companies think they are implementing OO, but they are actually busy with rapid prototyping using Delphi etc. It is only when the projects get longer that companies realise a solution cannot be hacked together.)

3.5 Soft Issues

It seemed as if people considered the soft issues such as the right attitude, communication etc. as being a bigger problem than the technical? Would you agree?

3.5.1 Resistance

Did the company have to deal with any resistance to OO? How did you deal with that?

3.5.2 Organisational issues

In half the cases studied in South Africa, the organisational structure has not changed at all. Various papers however mention the importance of such a change.

What is your opinion?

3.5.3 Management

During the local interviews, although it seemed as if management played an important role in the transition, there was no consensus whether the management of OO products is indeed different from the management of other projects.

What is your opinion?

3.6 First project

How would you describe the ideal first OO project? What is the experience in the USA?

(In South Africa the first OO project is typically 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's 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****. This suggests an* ***"in between"*** *project -- if* ***too small*** *it will not matter that 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 be inclined after a while to want to see results.*

3.7 Training

Regarding training -- do you have any suggestions?

What problems did you experience?

What is the state of the training organisations available? How do they rate?

In South Africa, there is a definite relationship between the progress in OO and the skill level of the developers. Still, companies' requirements

are generally not for people to be graduates or not, it is rather a case of people with practical experience that are in demand as well as people who fit into the company culture.

A South African retail company said that attitude (towards OO) is more important than aptitude. A small software company felt that people who are good in C, would also probably be good in C++.

Do you believe in re-training old developers or only employing new ones, using the existing group of developers for maintenance?

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?

3.8 Reuse

Do you have any guidelines for achieving reuse?

How does your company manage the retrieval of reusable objects? Do you have some kind of librarian who knows which objects exist?

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

too generic. The message was therefore concentrate on use rather than reuse. Most of these companies are however investigating appointing one person as a "reuse miner" to manage the libraries, plan the repository, etc.

3.9 Language

Which languages do you use? Why?

3.10 Legacy

The local interviews revealed that most companies had systems they have recognised as legacy systems. Interestingly, in quite a few cases these systems were relatively new (3 years), and some were even OO systems.[1]

What is your experience?

3.11 Testing

Which testing methods do you use or propose?

During the South African interviews it appeared as if testing did not receive much attention, due to either a lack of tools (for Smalltalk and Java), high costs involved with tools, or tools covering only limited sections of the system life cycle. A lack of user commitment (since in many of the cases the testing also needs to include an acceptance testing by the user) was also mentioned.

[1] Casais also referred to this problem in his article[15]

3.12 Quality

Did the pressure to get ISO 9000 play a role in the successful adoption of OO? Does the SEI (CMM) level play a role at all?

The results in South Africa show that experience in OO and quality seems to go hand in hand, even though many companies felt that quality was a good thing to have but didn't always have time for it. There is still not a lot of pressure for companies to get ISO 9000 in order to win a tender.

Would you agree? What is the experience in the USA?

3.13 Methodologies

Locally there is a relationship between experience in OO and the presence of methodology usage. Still, in many companies no methodology was used, highlighting again situations where new tools on the market were adopted and not the OO methodology per se.

What is your opinion and experience?

3.14 CASE

Regarding CASE tools, which did you use and which criteria did you use for choosing the tool?

Which problems did you experience with the CASE tools available?

Which levels of the System Development Life Cycle do you use methods and tools for? For example analysis, design, testing, etc?

In your opinion, what is the state of the tools available?

Local companies 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.

3.15 Metrics

What metrics did you use for tracking progress in the move towards OO? How did these metrics rate? Which tool do you use for data collection of statistics, time spent on projects, etc?

During the South African interviews, 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 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. In small companies due to priorities, there is no money available for metrics.

3.16 Timing

When did the company start getting involved in OO?

What factors influence the speed of adoption of OO?

Do you think it is easier to make the transition now? With new tools being available?

In South Africa there was no agreement. Some said that tools would not make the difference, others reason that the technology has now been tried and tested and that it is therefore easier.

3.17 What is next

Is OO still an emerging technology or has the technology matured?

What are the expected new developments?

What do you think of the move to components?

According to the Cutter Consortium report of 1997, most companies have started developing some form of component libraries, and 40% of these companies already have frameworks[38]

3.18 Problems

What were the kind of problems that you have experienced in the transition to OO? What mistakes typically exist?

3.19 Guidelines

Do you have any guidelines for companies who are thinking about making the transition to OO?



NDM QMS

NDM - Results From Overseas 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 3

1.1 Introduction 3

1.2 Audience 3

1.3 Applicable Documents 3

1.4 Requirements Traceability 3

1.5 Abbreviations 3

2 Overseas Interviews 3

2.1 Object-Z: Edmund Arranga 3

2.2 Object-Z: William Price 3

2.3 Objectgems: Dennis Laibson 3

2.4 The Travelers Group : Ron Calabrese 3

2.5 Presentation: Martin Fowler 3

Change History

Configuration Control

Project:	NDM QMS
Title:	NDM - Results from overseas interviews
Doc. Reference:	C:\MSC\TP\NDM312.100
Created by:	M Jansen Van Rensburg
Creation Date:	17 September 1998

Document History

Version	Date	Status	Who	Saved as:
0.01	98\09\16	Draft	MVR	NDM312.010
1.00	98\09\23	Approved	MVR	NDM312.100

Revision History

Version	Date	Changes
0.01	98\09\16	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 a record of the interviews held during September 1998 with various USA companies. A record of a presentation by an overseas speaker in South Africa on the issues concerned in the transition to OO is also included. These interviews were held to determine and compare the progress made in the transition towards Object Orientation in companies in South Africa and abroad.

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 Overseas Interviews

2.1 Object-Z: Edmund Arranga

The interview was held on 10 September 1998 in Long Beach, California. The company consists of 6 people, and activities include the publishing of books and newsletters related specifically to OO COBOL in interactive and distributed applications. The company also provides training in these areas.

The company was started in 1995. The reason for choosing OO COBOL was that there was already an abundance of literature available for C++, etc. and very little for object oriented Cobol.

OO was chosen for the classical reasons such as maintainability. Cobol was chosen because rules in Cobol make good objects.

Regarding market sector, there are more pressing needs in the financial sector for solutions, whereas the manufacturing sector lags, being more conservative.

Regarding company size, the opinion expressed was that one bad apple can spoil the OO effort in the whole company. Therefore small companies will be more successful. Arranga also referred to the book, The Mythical Man Month by Fred Brooks[13] (IBM) to motivate why small teams are more productive.

Regarding project size, he again prescribed a small team. For project lifetime he thought that huge cycles were undesirable.

For the first OO project he prescribed having relevance, high visibility and getting sponsorship with a lot of political influence. The transition should be done top down unless the company is very small.

He reasoned that it is the soft issues that outweigh the technical issues, especially in the move from Cobol to OO.

Regarding resistance, he would rather call it puzzlement. People ask the question: if they can do a certain task using structured Cobol, why do it any other way?

He warned that it is dangerous to draw a profile of the type of person to use in OO projects – "anybody with an open mind and right attitude can succeed".

Regarding company size he had found that the transition is more successful in small companies where people typically wear different hats and not only do certain tasks.

On the issue of management he had found that OO requires a different mindset due to different constraints. Managers need to know the technology and terminology. Management is a weak area and people often step back due to the fear of failure. Managers should be mentored.

Regarding training, he teaches at UCLA (University of California Los Angeles) where OO is merely a subtopic of software engineering. He prescribes first teaching the language but not in too much detail. People need to believe what they learn is a better technique.

Ideally, OO should be taught in an iterative way just as the OO development process is iterative. It has to be evolutionary. This makes the transition hard to implement since people tend to hold on to what they know.

Regarding the state of training: universities lag behind industry although private universities are slightly better. The pace of change is just too fast to update the curriculum. There is no overall course teaching all the elements together.

Huge management is required to achieve reuse - components are going to be the unit of reuse in the future.

Regarding legacy systems he prescribes wrappers and "leave as is" and therefore only examining new functionality. In their company they are currently working on strategies for using distributed object calls to the legacy systems.

According to him, normally legacy systems equate to being critical systems.

Regarding quality improvement, ISO 9000 plays no role since the USA will always oppose what Europe prescribes. The principles of the CMM (the Capability Maturity Model) is sometimes enforced. He is against the idea of companies first moving to CMM level 3 and then to OO.

He uses a combination of UML and Rebecca Wirfs-Brock's methodology as a methodology.

Arranga found that there are many cases where people only use the OO tools and think that they are busy with the transition to OO.

Generally they do not use any CASE tools, (some Rational though), mostly not because there is no support for Cobol.

Regarding the state of tools he thought that there has been lots of improvement and that tools are more flexible now.

Regarding metrics, the company does not use any. The comment was that "people are weary of metrics because of the great potential for misuse. Generally they are satisfactory but you have to train management to use them"

The factor influencing the speed of adoption is commitment.

He thought that the adoption of OO should be easier now since the dust has settled, in standards and notation. There are lots of resources now, and OO is not considered a fad anymore. Five years ago there were 20 techniques available which made it difficult then.

On the question whether the technology is still emerging, he commented that compilers are not mature yet -- new versions are still made available. The technology is immature and is still new.

New developments will be in components as well as distributed objects that live on the web.

2.2 Object-Z: William Price

The interview was held on 11 September 1998 in Orinda, California.

Price teaches at UCLA and presents workshops on OO Cobol and web-Cobol. OO was chosen for the classical reasons such as maintainability. He mentioned as example a case where a company's first project took 8 months and the second 6 weeks, indicating a huge improvement in reuse.

Regarding company size he felt that large companies and small companies have both successfully implemented OO, so no conclusions can be drawn.

Regarding project size he suggested a small contained application that finishes quickly.

His recommendation for project lifetime is that a project should not drag on.

Regarding resistance he reasoned that Cobol developers are slow to change (he used the 1987 Datamation survey that included questions about a change to Cobol 85 to motivate his opinion that people clearly did not appreciate the technical aspects of programming). He had also found that not many Cobol programmers know other languages or programming theories. The backgrounds are different for Cobol versus C programmers.

He said that management's role is important for 2 reasons:

- releases are now easier because components can be plugged into a new application with little effort.. OO provides this ability and managers need to know about it.
- A greater part of the time spent will be on analysis and design which becomes complicated.

Regarding training he found that problems exist in the design arena – learning the language is easy.

The small number of people that have recently attended workshops he held shows that a lot of people are currently involved in solving the Year 2000 problems, thereby slowing down the adoption of OO.

Regarding programmers he found that there are always people who do not want to change or who do not manage.

He found that more training is necessary for people involved in the design of OO systems, than for the actual developers ("coders")

He had found that it can take less than 8 to 9 months to complete the transition to OO.

Regarding reuse he found that the value of reuse is overstated. Reuse is hard to achieve, you need a lot of insight into the system being developed and the actual business system is complicated.

When training, he suggested looking at what is necessary in the future, rather than the present. There are still numerous structured systems needing maintenance at present but future systems will be object oriented and therefore newcomers should be trained to handle the future systems effectively.

He reasoned that C should not be a prerequisite for C++ training.

For testing he suggested breaking the system into compartments. Since there should now be no coupling involved in OO systems, it should be easy to test all cases.

He uses the UML methodology.

Regarding CASE tools, he does not use any. His comment "10 years ago that was going to solve all the problems" confirmed opinions found in various articles.

No metrics are used, the reason being that people are too busy.

Factors influencing the speed of adoption is dedication and management.

On the question whether it would be easier now to adopt OO, his comment was the following:

"If you were first and you failed everybody saw your failure. The bad way to think is that the longer you wait the safer it gets. The sooner you start, the better, since you have to be quick to market. It all depends on the competitiveness of the business."

He thought that OO is still an emerging technology and will be until people have experience in the technology. (OO Cobol specifically)

The next developments for OO will be CORBA-related, and will include web objects, transparency and integration between objects. In OO-Cobol specifically there will be standardisation of class libraries.

2.3 Objectgems: Dennis Laibson

The interview was held on 9 September 1998 in McLean, Virginia. The company consists of 12 people. The company is considered to be a so-called "boutique firm" (small and specialised) and is an authorised IBM partner. They have an object connection program, provide training in Smalltalk, develop OO applications and also provide consulting.

Regarding languages they use Smalltalk and now also Java.

They started with OO in 1995, the reasons for OO being maintainability - they also saw that there was more demand for OO than supply.

Regarding market sector, the majority of their clients are from the financial sector and therefore the conclusion that the financial sector has progressed to OO faster must be correct.

Regarding company size, he reasoned that large companies would do better since OO requires a corporate decision and large investment.

This also applies to project size (for the same reason as above).

Regarding resistance: no resistance to OO was experienced, but the concept is often loosely interpreted.

Regarding organisational structure, he did not believe that there needs to be any change since the same rules apply.

Regarding management, there is no need for change - as before it is a case of knowing the cause, impact and future.

His recommendation for the first OO project, is that one should complete a small project or part of a project first. The reality is that there are often time constraints involved which does not allow time for experimentation.

For training he suggests first learning the OO concepts and then the language.

Regarding the type of people to use, he believes that you cannot teach an old dog new tricks, therefore new people do better.

When choosing languages, unfortunately marketing is the important factor. Smalltalk is technically better than Java but Java is more popular.

Regarding the state of training, the company only employs developers with at least 3-5 years experience.

In terms of quality improvement, Laibson saw no requirement for this in the USA.

The state of methodologies is satisfactory – his company uses a combination of more than one.

Visual Works (the tool they use) has become so powerful that they do not need CASE tools. Therefore they do not use any CASE tools.

Regarding the factors influencing the speed of adoption of OO, he found that the choices are all dollar driven.

The adoption of OO now should be easier. Debugging has taken place so that the tools are more bug free now. He reasoned that if a team does their homework, they would talk to people and therefore not make the same mistakes as previous companies did.

General Problems / guidelines: he suggests using a wise architect to do a true evaluation, as well as investing in architecture first.

2.4 The Travelers Group : Ron Calabrese

The interview was held on 8 September 1998 in Hartford Connecticut.

The company consists of 20000 people.

Activities include support, development and research in the insurance area.

The adoption of OO started in 1992, with a major application. OO was chosen not as much for reuse as for handling complexity. They started with a large application - it however helped to be visible since at times they were tempted to give up.

Regarding market sector, the opinion was that not many companies are doing OO correctly, even though many think they are. According to an MIT case study[50], investigating the company's adoption of OO, claims processing was 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.

Regarding company size it was felt that smaller is better; they had 30 developers.

For project lifetime, the experience was that with development, you couldn't see much progress until the project is completed which supports why the project lifetime should be short.

They developed frameworks 4 years before which was advanced for the time. However, these frameworks were proprietary which made reuse difficult.

Regarding resistance Calabrese thought that it existed but not in their company. He thought that a possible explanation was that the manager knew OO well and therefore did not fear the unknown.

They had a separate technical and project manager.

Regarding organisational structure, he saw a definite need for change. They had a small group that handled the technical structure and then smaller teams to handle the business components.

Regarding management, he found that if you use components there is no need for change in the management techniques, since the system is divided into smaller tasks that are handled separately. They used traditional project management techniques with their new components-based architecture.

The first project was critical and had high profile - it seemed to him to be that in most cases.

Regarding training, they found it hard to find people with OO experience. They partnered with a training company and taught a specific methodology, and also catered for courses in debugging classes, C++, code walk throughs, and mentoring.

Old versus new developers: they used both Cobol developers and others. Some developers made the shift towards OO successfully and others not. It seemed to depend on the individual's capabilities.

They were very successful in achieving reuse - perhaps not in the first project but definitely in the second and third.

C++ was the language of choice 5 years before - since people were familiar with and had experience in C it was a natural choice when moving to OO.

Regarding testing they had test drivers per component and per framework. They have a laboratory for automated regression testing and testing across operating systems.

Quality (ISO 9000, SEI CMM levels etc.) does not play any role in the company.

The company uses its own methodology which is a Rumbaugh / Booch combination based on their architecture, that works well. They also use multiple iterations.

No CASE tools are used - the tools are limiting, and there is no flexibility from a specific vendor.

The factors influencing the speed of adoption is knowing that there has been success in other groups.

The next step for OO is standards for distributed OO.

Regarding general problems: they experienced business rather than technical problems. They thought they could do without consultants but couldn't. Too big a group is undesirable, while daily reviews are important.

Their reuse directive was as follows:

- to establish one divisional architecture
- not to over engineer any component

- creating common functions and pluggable components

For the frameworks they had two kinds of developers:

A Business developer who

- develops use cases,
- determines component services
- develops screens
- develops business objects
- tests components

A Technical developer who

- concentrates on technical aspects
- designs and develops base classes
- mentors business developers

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

Their technological achievement was 100% reuse of framework components and foundation classes.

The organisational structure changed as follows:

1992: 1 large development team (24 developers, including 10 consultants, project managers, 1 senior OO consultant,1 client server engineer

1997: multiple small development teams (3-5 per team), project management done by the team

The emphasis in 1992 was:

- Reengineer the business and learn OT
- Providing new functionality quickly

In 1997 it was:

- Still providing new functionality quickly
- Promoting a flexible architecture
- Reuse wherever possible

The following valuable information was obtained from the MIT case study[50] and proved useful in the further understanding of the company's transition to OO:

"Four kinds of obstacles have prevented successful implementation of OO environments in individual firms."

The Travelers group experienced all four:

- 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 in the "not invented here" bias) and
- political difficulties (securing funds was required since building for reuse can cost more)

The started OO on large-scale system without prior OO knowledge and limited distributed experience.

The tools were immature, but were rapidly improving.

Regarding reuse, 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" was the comment.

They were missing milestones and had to relax reuse requirements to keep up.

"All the prior aggravation paid off" said the project manager

A three-tier model was developed which was later reused on new projects. Reuse also took place in terms of learnt skills and the centrally provided support infrastructure.

The new model had business objects at the top, with parallel objects in the second layer (called Frameworks). These frameworks acted as object request broker for the queuing of requests. The bottom tier was platform objects that interfaced to specific platforms.

The most benefit from OO came in the development time of new systems.

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.

Their strategy for handling the four problems mentioned was:

- conceptual difficulties: (the paradigm shift) -- do modelling and coding as an iterative process, since the more objects you develop the easier it becomes.
- Technical problems: develop in-house expertise, create centralised support, develop a 3 tier architecture to allow developers to specialise on a limited number of technologies
- Organisational problems: reuse expertise, use the three-tier architecture to understand how the objects will be used
- Political problems: communication with business partners

They learned the following lessons:

1. OO development in the pure sense is not practically implementable.- business needs change too quickly so that the time required to fully develop the OO model will never be available.

2. Accept that technologies are unstable and changing – this emphasises the need to have objects that allow you to change between platforms easily

3. Expect pain (new systems and immature technologies requires adjustment to organisational and technical problems simultaneously) but provide rapid relief

4. Adopt a learning attitude that recognises the need to learn from mistakes

5. Design a reliable cost effective [illegible]cture for supporting the OO environment

6. Hand over ownership for technology problems – no manager can understand all the technologies that are required

These lessons apply to any IT environment that relies on an immature or fast changing technology.

Development cost decreased from $7million (1993) to $0.5 million (1996).

2.5 Presentation: Martin Fowler

Fowler gave a presentation on his experience in the transition to OO at Chrysler, on 27 May 1998 in Rosebank at the Park Hyatt hotel as part of an OTSIG[1] event.

He described a payroll system, where people moved from COBOL to OO. The IT manager's reason for using OO was flexibility and not reuse.

Fowler said that "if reuse is the reason think again."

Due to insufficient funding, they used experiments to help in the learning curve.

They brought the developers physically together which was important.

They had to choose between Smalltalk and C++ and chose Smalltalk for memory management

[1] OTSIG is an acronym for the Object Technology Special Interest Group, based in Johannesburg and Cape Town, South Africa.

An important success factor was separating training and development.

Development started in 1994. Early in 1995 there were the following danger signs:

- People believed the system was simple
- There was a mismatch between Cobol and Smalltalk developers.
- The team did not believe in testing until the last month
- They kept looking for general solutions and did too much abstraction.
- The system was patched – as it became more complex it became increasingly difficult to add a function

The system had to be restarted – they did the following right the second time:

- 2 people worked on one work station
- they built self testing code
- they integrated everyday and tested – the importance of continuous integration was mentioned
- they developed use cases
- they used incremental development
- the development was cut from 35 to 16 people
- they fixed problems as soon as they appeared
- They used refactoring (extreme programming) to achieve reuse.

Author: Janzen van Rensburg

Name of thesis: Pitfalls and guidelines in the transition to object oriented software design methodologies

PUBLISHER:

University of the Witwatersrand, Johannesburg
©2015

LEGALNOTICES:

Copyright Notice: All materials on the University of the Witwatersrand, Johannesburg Library website are protected by South African copyright law and may not be distributed, transmitted, displayed or otherwise published in any format, without the prior written permission of the copyright owner.

Disclaimer and Terms of Use: Provided that you maintain all copyright and other notices contained therein, you may download material (one machine readable copy and one print copy per page)for your personal and/or educational non-commercial use only.

The University of the Witwatersrand, Johannesburg, is not responsible for any errors or omissions and excludes any and all liability for any errors in or omissions from the information on the Library website.