Software Engineering Practices in Singapore

Danny C.C. Poo and Mui Ken Chung
Department of Information Systems
School of Computing
National University of Singapore
3, Science Drive 2
Singapore 117543
e-mail : dpoo@comp.nus.edu.sg <Danny Poo>

Abstract

A study on Software Engineering practices in Singapore was conducted in 1995. The study was conducted to gain an insight into the extent to which Software Engineering practices have been adopted by organisations in Singapore, the benefits these organisations have realised, the problems they face in implementing these practices and their perceptions on issues prevalent in the Software Engineering field. In the context of the study, "Software Engineering Practices" refer to the models, techniques and tools used in the Software Engineering process.

The study was conducted through a self-administered mail questionnaire survey sent to a number of organisations in Singapore. A total of 54 organisations responded to the survey. The results showed that the majority of organisations had adopted a formal system development methodology but relatively few were using Software Engineering tools. The respondents' perceptions towards the benefits and problems of implementing Software Engineering practices validated to a large extent the findings of other practitioners and researchers in the field.

1.0 Introduction

According to Conger (1994), Software Engineering is the systematic application of tools and techniques in the development of computer-based applications with the aim to solve problems. Many have suggested that the application of Software Engineering techniques could alleviate the problems experienced by developers e.g. spiralling software costs, inconsistent software quality, increased software maintenance burden (Fletcher and Hunt, 1993).

Although organisations in Singapore have applied Software Engineering techniques in their system development projects, it is not clear how much or how far have the techniques been practised in Singapore. To gain an insight into the extent to which Software Engineering practices have been adopted by software developing organisations in Singapore, the benefits these organisations have realised, the problems they face in implementing these practices and their perception on issues prevalent in the Software Engineering field, a survey on Software Engineering practices was conducted in Singapore in 1995.

In the context of the study, "Software Engineering Practices" include methodologies, models, techniques and tools used in the Software Engineering process. We shall follow the definitions given by Fletcher & Hunt (1993) on methodologies, models, techniques and tools. A methodology is a systematic series of actions integrating a set of complementary techniques to build and maintain systems; for example, James Martin's Information Engineering, Ernst & Young's Navigator Systems Series, Arthur Andersen's Method 1, etc. Models are integrated assertions, theories, concepts and strategies that constitute a way of developing systems; for example, waterfall model or variations of it to produce software. Techniques are specific set of actions to accomplish a given activity; for example, Data Modelling, Data Flow Diagrams, Joint Application Development, etc. Tools are products which assist a software engineer in developing and maintaining software; for example, CASE, testing and application front-end development tools.

1.1 Purpose of Study

An understanding of the current Software Engineering practices in Singapore would allow us to benchmark ourselves against the state-of-the-art in the Software Engineering field. Such a comparison would, hopefully, show up areas where we have performed adequately and areas where there is room for improvement, so that energies and resources can be targeted more specifically at areas where "catching up" needs to be done.

1.2 Objectives of Study

A survey on the use of software development methods in Singapore was earlier conducted by Phoon and Poo (1991) in 1991. Subsequently in 1993, another survey on software project management practices in Singapore was conducted by Khor and Woo (1994). The study reported here hopes to supplement the aforementioned studies and achieve the following objectives:

a. To find out the extent to which Software Engineering practices are carried out in Singapore.
b. To identify the most commonly used software process models, techniques and tools.
c. To identify factors which influence Software Engineering practices.
d. To identify the problems organisations face in adopting Software Engineering practices.
e. To gain an insight into organisations' perception towards Software Engineering practices.
f. To gain an insight into the nature of software maintenance work in Singapore.

This paper discusses the results of the survey. Due to space constraint, this paper will focus on the first five objectives (a-e). Objective f will be reported in a separate paper.

1.3 Organisation of Paper

This paper is organised in the following manner :

Section 2 :	Discusses the study methodology.
Section 3 :	Summarises the results of the survey.
Section 4 :	Analyses the results of the survey in relation to the objectives and hypotheses formulated for the study.
Section 5 :	Discusses the implications of the study.
Section 6 :	Concludes the discussion.

2.0 Study Methodology

This section discusses the methodology used to conduct the study. In particular, it will discuss the following :

- Objectives and hypotheses.
- Unit of Analysis.
- Research variables.
- Research sample.
- Survey instrument.

2.1 Objectives and Hypotheses

This study attempts to investigate whether the following hypotheses, formulated based on the study objectives, are true:

Objective (a):

To determine the extent to which Software Engineering practices are carried out in Singapore.

Hypothesis 1:

Software Engineering is practised in the majority of organisations in Singapore.

Rationale : In the earlier study conducted by Phoon and Poo (1991) it was found that the majority of organisations in Singapore used a formal system development methodology. This hypothesis is to test whether a similar observation can be made on Software Engineering practices in this study.

Hypothesis 2:

Software Engineering techniques are more widely used than Software Engineering tools (due to historical and cost reasons).

Rationale: Software Engineering tools are developed in support of Software Engineering techniques and are relatively more expensive to implement than Software Engineering techniques. This hypothesis is to test whether these factors have affected the extent of usage of Software Engineering tools as compared to Software Engineering techniques.

Objective (b):

To identify the most commonly used software process models and techniques.

Hypothesis 3:

The sequential/waterfall model is the most commonly used software process model.

Rationale: Since the sequential/waterfall model was amongst the first software process models to be developed, this hypothesis is to test if it is the most commonly used model.

Hypothesis 4:
The percentage of organisations which are currently using object-oriented techniques is relatively low.

Rationale: Since object-oriented techniques are relatively new, it would be logical to assume that their usage has not reached a significant level among organisations yet. This hypothesis is to test whether this assumption is true.

Hypothesis 5:
Software Engineering techniques and tools are used more extensively in the analysis and design phases of the software production process.

Rationale: In the study conducted by Phoon and Poo (1991), it was found that the use of system development methodology was most extensive in the analysis and design phases. This hypothesis is to test whether a similar observation can be made for Software Engineering techniques.

Hypothesis 6:
Some techniques are more commonly used than others.

Rationale: The aim of this hypothesis is to find out which techniques are more commonly used in each phase of the software life cycle.

Objective (c):
To identify factors which influence Software Engineering practices.

Hypothesis 7:
The adoption of Software Engineering practices is influenced by:

a. size of organisation
b. industry sector/type

Rationale: It is usually assumed that Software Engineering practices would be more prevalent in larger organisations due to the higher complexity of systems developed and also the availability of resources to invest in implementation of Software Engineering practices. It is also assumed that certain industries, due to the nature of the systems developed to support the business, would have a higher percentage of Software Engineering usage. This hypothesis is to test whether these assumptions are true.

Hypothesis 8:
Formal training has a positive effect on the adoption of Software Engineering practices.

Rationale: In theory, formal training heightens awareness of the benefits of Software Engineering practices and enables the proper application of Software Engineering techniques to realise these benefits. This hypothesis is to test whether this is true in practice.

Hypothesis 9:
Organisations where the IT function has a high profile tend to implement Software Engineering practices more extensively.

Rationale: In theory, top management support is one of the predominant factors which affect successful adoption of Software Engineering practices. This hypothesis is to test whether this is true in reality.

Hypothesis 10:
The adoption of a Software Engineering methodology forms the basis for use of Software Engineering techniques and tools.

Rationale: Software Engineering techniques and tools were developed to support Software Engineering methodologies. This hypothesis is to test whether in practice, there is correlation between the use of Software Engineering techniques and tools and the adoption of a Software Engineering methodology.

Objective (d):
To identify the problems organisations face in adopting Software Engineering practices.

Hypothesis 11:
There are common problems faced by organisations in adopting Software Engineering practices.

Rationale: Problems faced by organisations in adopting Software Engineering practices have been well documented. This hypothesis is to test whether these problems are also faced by the organisations surveyed.

Objective (e):
To gain an insight into organisations' perception towards Software Engineering practices.

Hypothesis 12:
In general, organisations have realised benefits with the implementation of Software Engineering practices.

Rationale: The benefits that organisation will reap for implementing Software Engineering practices have been well documented. This hypothesis is to test the perception of organisations surveyed towards the realisation of these benefits.

Hypothesis 13:
Software Engineering practices have a positive effect on measures of software development success.

Rationale: The primary objective of Software Engineering is to improve the software development and maintenance process. This hypothesis is to test the perception of organisations surveyed towards the impact of Software Engineering on their software development process.

2.2 Unit of Analysis

The unit of analysis was the organisation. The target representative for the unit of analysis was the IS department in the case of IT user organisations, the software development group in the case of IT vendors and the systems consultancy group in the case of IT consultancy firms. These representatives of the unit of analysis were selected because they were most likely to have experience in Software Engineering within the organisation.

The unit of observation was the employee of the organisation who responded to the survey; the employee is either a software development manager or executive.

2.3 Research Variables

The nature and sophistication of Software Engineering practices were measured by the following variables:

- Software Development Methodology
  Existence, nature and extent of software development methodology used.
- Software Engineering Techniques

Software Engineering techniques used in each phase of the software life cycle.

- CASE Tools
  Existence and extent of usage of CASE tools.
- Training and Education in Software Engineering
  Existence and type of training and education provided in Software Engineering.

Perceptions of current Software Engineering practices were gathered by the following variables:

- Measurement of Success in Software Projects
  Frequency of projects being completed on time, within budget, meeting user requirements and easy to maintain.
- Factors Influencing the Success of Software Engineering Practices
  Extent to which organisations agree/disagree with commonly cited factors which contribute to the success of Software Engineering practices.
- Barriers to the Successful Adoption of Software Engineering Practices
  Extent to which organisations agree/disagree with commonly cited factors which prevent the successful adoption of Software Engineering practices.

The research variables identified above were operationalised in the form of one or more survey questions with closed or open-ended answers. The survey form was designed with a conscious effort to minimise the use of open-ended questions so as to make it easier for respondents to fill in the questionnaire and to facilitate analysis of data.

Answers to the questions were gathered in the form of either nominal, ordinal or interval scales. A *nominal scale* is one where the numbers are used merely as labels and the size of the number is meaningless e.g. 1 is used to denote male and 2 is used to denote female. An *ordinal scale* is one where there is some correspondence between the size of the number and the order of magnitude of the quality represented by the numbers; however, the actual size of differences in magnitude cannot be gauged from the scale e.g. the position 1, 2 and 3 in a race tells who came first, second and third in the race but does not indicate the difference in speed of the three persons. In an *interval scale*, the numbers do represent the magnitude of the differences e.g. numbers on a temperature scale.

2.4 Research Sample

A sample of 240 organisations was selected from the mailing lists used in past surveys and personal contacts. The sample of organisations selected included IT users from both the private and public sector and IT vendors. A total of 54 valid responses (giving a response rate of 22.5%) were received.

2.5 Survey Instrument

A self-administered survey form consisting of 65 multiple choice and 6 open-ended questions was used. The questionnaire was structured as follows:

Section 1	-	Demographic Data
Section 2	-	Current Software Engineering Practices

- Software Development Methodology
- Software Engineering Techniques
- CASE Tools
- Software Maintenance
- Training and Education In Software Engineering

Section 3	-	Perception of Current Practices

3.0 Results of Survey

This section presents the results of the survey. Where longitudinal comparisons are to be made, results from the earlier studies conducted by Phoon and Poo (1991) and Khor and Woo (1994), are also shown in shaded portions.

3.1 Demographic Data

All the three industry sectors i.e. the government organisations, private local organisations and private foreign-owned organisations, were well represented in the survey. The majority of the respondents were senior IS managers and executives.

IT developers formed the most significant group of respondents at 66.7%, followed by software suppliers at 11.1%, training/education/R&D organisations at 9.3% and IT consultants at 7.4%. The remaining 5.6% of respondents were hardware suppliers and others.

The largest percentage of organisations who responded had between 100-499 employees (31.5%). There were also a significant number of organisations with either less than 100 employees or more than 2000 employees, each forming 22.2% of the respondents.

Most (40.7%) of the organisations had less than 10 IT professionals in the organisation. 25.9% had between 10 to 19 IT professionals, 14.8% had 20 to 49 IT professionals and the remaining 18.5% had more than 50 IT professionals.

29.6% of the respondents had developed between 1 to 5 software systems in the past five years, 27.8% had developed 6 to 10 systems, 13% had developed more than 20 systems. Operational systems which support day-to-day running of the organisation formed the bulk of software systems developed by the organisations, followed next by decision support systems which facilitate management decision making, then inter-organisational systems which provide links to business partners and finally, other types of systems. Developing new applications was the main activity of the IS departments, followed by maintenance of applications as the second most common activity while end user computing support together with technical operations support tied for third place in the ranking of activities carried out.

3.2 Current Practices - Software Development Methodology

68.5% of the organisations which responded used a formal system development methodology (see Table 1). 78.4% of the formal system development methodologies used were developed in-house. Since system development methodology and project management methodology are both Software Engineering practices, these findings were compared against those obtained in the study conducted by Khor and Woo (1994), where 61.6% of respondents reported having a formal project management methodology, of which 71% were developed in-house and 60.8% of the organisations had been using their methodology for more than 3 years (see Table 1). This comparison shows that there is a close correspondence between the usage of formal system development methodology and project management methodology.

Organisation Uses Formal System Devt Methodology	**Response**	**Freq**	**Percent**	**Freq**	**Percent**
	Yes	37	68.5%	61	69.0%
	No	16	29.6%	27	31.0%
	(Missing)	1	1.9%		
	Total	54	100.0%	88	100.0%

Table 1 : Organisations using Formal System Development Methodology

86.5% of these organisations had been using the formal methodology for at least a year, with 27% using it for between 1 and 2 years, and 59.5% of these having used it for at least 3 years (see Table 2**).**

How Long Has Methodology Been In Use ?	**Response**	**Freq**	**Percent***
	Less than 1 year	5	13.5%
	1 - 2 years	10	27.0%
	3 - 4 years	7	18.9%
	More than 4 years	15	40.5%
	Total	37	100.0%

* 37 Cases = 100% (based on number of respondents who used a formal system development methodology as shown in Table 1)

Table 2 : Number of Years Formal System Development Methodology Has Been Used

Taking into consideration both formal and informal methodologies used, the largest percentage (37%) of methodologies used followed the Sequential/Waterfall software process model. The next most common model followed was the Incremental Model (Phased Delivery) at 16.7%.

The Rapid Prototyping Model was followed by 9.3%, the Spiral Model (Waterfall + Risk Analysis) by 3.7%. 11.1% were not sure which model was followed - not surprising since this question requires some academic background in system development methodologies and some of the respondents may not be from an IT discipline. The result is shown in Table 3.

	Response	**Freq**	**Percent***
Software Process Model Followed by Methodology	Sequential/Waterfall Model	20	37.0%
	Spiral Model (Waterfall + Risk Analysis)	2	3.7%
	Incremental Model (Phased Delivery)	9	16.7%
	Rapid Prototyping Model	5	9.3%
	Others	1	1.9%
	Not Sure	6	11.1%

*54 Cases = 100%

Table 3 : Software Process Model

The wide application of the Sequential/Waterfall model is expected since it is one of the earliest software process models developed and is largely consistent with the structured analysis, design and programming methods introduced in the 1970s. It is familiar to many IT professionals and has been used to develop a whole continuum of application types ranging from transaction-based to real-time systems (Degrace and Stahl, 1990).

3.3 Current Practices - Software Engineering Techniques

Requirements Analysis

Table 4 shows the most commonly used techniques in requirements analysis. Half of the respondents used the following 4 techniques for requirements analysis :

- Data Flow Diagrams (77.8%)
- Data Dictionary (57.4%)
- Entity Relationship Diagrams (51.9%) and
- Flowcharts (51.9%)

The other more commonly used techniques were data normalisation (46.3%), prototyping (42.6%) and process decomposition diagrams (25.9%). Object-oriented analysis and entity life cycle analysis were the least used techniques with 9.3% response each.

According to Keyes (1992), the entire object-oriented field i.e. object-oriented analysis, object-oriented design and object-oriented programming, is in the same state of turbulent evolution that the structured field underwent in the mid-'70s. Object-oriented techniques reflect a new culture of Software Engineering, one of component culture, as compared to the project culture of the more traditional techniques (Mandrioli and Meyer, 1992). As such, it is not surprising that very few organisations have yet to venture into the object-oriented field, and of those who have, many are testing it out on small scale projects first. The usage of object-oriented techniques in the United States around 1994 was only about 5% and there was no usage of object-oriented techniques in Hong Kong based on the results of the survey on software maintenance conducted by Yip, Lam and Chan (1994). In a recent survey conducted by Applied Computer Research and Computing Trends in United States, Glass (1996) reported that Information Systems (IS) organisations are not terribly interested in the object-oriented approaches.

	Rank	Response	Freq	Percent*
Techniques Used in Requirements Analysis Phase	1	Data Flow Diagrams	42	77.8%
	2	Data Dictionary	31	57.4%
	3	Entity Relationship Diagrams	28	51.9%
	3	Flowcharts	28	51.9%
	4	Data Normalisation	25	46.3%
	5	Prototyping/Rapid Prototyping	23	42.6%
	6	Process Decomposition Diagrams	14	25.9%
	7	Process Dependency Diagrams	10	18.5%
	8	Joint Application Development (JAD)	9	16.7%
	9	Entity Life Cycle Diagram	5	9.3%
	9	Object Oriented Analysis	5	9.3%
	10	Others	2	3.7%

*54 Cases = 100%

Table 4 : Most Commonly Used Techniques in Requirements Analysis Phase

System Design Phase

The three most common system design techniques used were screen design (79.6%), report design (68.5%) and structure charts (40.7%) (see Table 5). Techniques which were used by 20% to 25% of the respondents were decision tables/trees, process hierarchy diagrams, HIPO diagrams and transaction volume analysis. Object-oriented design was the second least used system design technique at 3.7%.

	Rank	Response	Freq	Percent*
Techniques Used in System Design Phase	1	Screen Design	43	79.6%
	2	Report Design	37	68.5%
	3	Structure Chart	22	40.7%
	4	Decision Tables/Trees	14	25.9%
	5	Process Hierarchy Diagrams	13	24.1%
	6	Hierarchical Input Process Output Diagrams	12	22.2%
	7	Transaction Volume Analysis	11	20.4%
	8	Action Diagrams	8	14.8%
	9	Dialogue Flow Diagrams	4	7.4%
	9	Others	4	7.4%
	10	Object-Oriented Design (e.g. Booch Diagrams)	2	3.7%
	11	Warnier-Orr Diagrams	1	1.9%
	12	Nassi-Schneiderman Diagrams	0	0.0%

*54 Cases = 100%

Table 5 : Most Commonly Used Techniques in System Design Phase

Coding Phase

For this phase, program specifications and 4GL had the highest usage at 61.1% and 63% respectively. 38.9% of respondents still used 3GL (see Table 6).

	Response	Freq	Percent*
Techniques Used in Coding Phase	Program Specifications	33	61.1%
	4GL	34	63.0%
	3GL	21	38.9%
	Others	4	7.4%

*54 Cases = 100%

Table 6 : Techniques Used in Coding Phase

Review/Testing

The most widely used review technique was walkthroughs at 72.2% as shown in Table 7. Following at a distant second was black-box testing at 42.6%. Bottom-up testing and top-down testing were practised by 27.8% and 25.9% of the respondents respectively. White-box testing was carried out by 18.5% of the respondents.

	Rank	Response	Freq	Percent*
Techniques Used in Review/ Testing	1	Walkthroughs	39	72.2%
	2	Black-box Testing	23	42.6%
	3	Bottom-up Testing	15	27.8%
	4	Top-down Testing	14	25.9%
	5	White-box Testing	10	18.5%
	6	Others	1	1.9%

*54 Cases = 100%

Table 7 : Techniques used in Review/Testing Phase

In white-box testing, test cases are based on program codes rather than functional specifications (which is the basis for black-box testing). The different forms of white-box testing include statement, branch and path coverage i.e. running a series of tests to ensure that all statements, branches and paths are tested at least once. In all these forms of testing, a tool is usually needed to help the tester keep track of the statements, branches and paths which have been covered (Schach, 1993). The complexity of white-box testing may be the reason why it is the least used testing technique.

Maintenance Phase

The most common Software Engineering practice instituted for the maintenance phase was software change management procedures (72.2%), followed by documentation change management procedures (57.4%) and finally configuration management procedures (33.3%) (see Table 8).

	Rank	Response	Freq	Percent*
Techniques Used in Maintenance Phase	1	Software Change Management Procedures	39	72.2%
	2	Documentation Change Management Procedures	31	57.4%
	3	Configuration Management Procedures	18	33.3%
	4	Others	0	0.0%

*54 Cases = 100%

Table 8 : Techniques Used in Maintenance Phase

Number of Techniques Used

On average, a respondent's organisation used 4 Software Engineering techniques in the requirements analysis phase, 3 in the system design phase and 1 each in the coding phase, review/testing phase and maintenance phase. This is shown in Table 9. This indicates that Software Engineering techniques are most widely practised during the requirements analysis and system design phases.

	Phase	Mean	Mode	S.D.
No. Of SE Techniques Used In Each Phase	Requirements Analysis Phase	4.111	4	2.597
	System Design Phase	3.167	3	1.969
	Coding Phase	1.704	1	0.964
	Review/Testing Phase	1.899	1	1.327
	Maintenance Phase	1.630	1	0.938

Table 9 : Number of Techniques Used in Each Phase

3.4 Current Practices - Training and Education In Software Engineering

51.9% of respondents indicated that their organisations provided formal training in Software Engineering methods (see Tables 10 and 11). Of these, 75% of them had courses conducted by external trainers and 39.3% had courses conducted by in-house trainers. Out of all the respondents, 46.3% had on-the-job-training in Software Engineering methods and only 5.6% were self-taught through videos, computer based training etc.

Does Your Organisation Provide Formal Training In Software Engineering Methods?	Response	Freq	Percent
	Yes	28	51.9%
	No	23	42.6%
	(Missing)	3	5.6%
	Total	54	100.0%

Table 10 : Training in Software Engineering Methods

3.5 Measures Of Software Systems Developed

From the data shown in Table 12, it can be seen that of the four measures used to gauge the success of software development projects in the organisations surveyed, the measure "Software implemented meets with a high degree of user satisfaction" scored the highest with a mean of

3.648 (out of a Likert scale of 1 to 5), indicating that organisations usually had no problems in meeting this goal. This is followed in order of decreasing mean scores by "Software developed are easy to maintain" with a mean score of 3.396, "Software projects are completed within estimated cost" with a mean score of 3.377 and lastly, "Software projects are completed on schedule" with a mean score of 3.333.

	Rank	Response	Freq	Percent*
Forms Of Training		**Formal Training**		
	2	Courses conducted by external trainers	21	38.9%
	3	Courses conducted by in-house trainers	11	20.4%
		Informal Training		
	1	On-the-job training	25	46.3%
	4	Self-taught (through videos, CBT, etc.)	3	5.6%
	5	Others	0	0.0%

*54 Cases = 100%

Table 11 : Types of Training

	Rank*	Software Situation	Response**	Percent***	Mean	S.D.
Frequency Of Occurrence Of These Software Situations	1	Software implemented meets with a high degree of user satisfaction.	Rarely/Never	5.6%	3.648	0.756
			Sometimes	35.2%		
			Usually	48.1%		
			Always	11.1%		
			(Missing)	0.0%		
	2	Software developed are easy to maintain.	Rarely/Never	11.2%	3.396	0.793
			Sometimes	38.9%		
			Usually	44.4%		
			Always	3.7%		
			(Missing)	1.9%		
	3	Software projects are completed within estimated costs.	Rarely/Never	18.5%	3.377	0.882
			Sometimes	31.5%		
			Usually	40.7%		
			Always	7.4%		
			(Missing)	1.9%		
	4	Software projects are completed on schedule.	Rarely/Never	18.5%	3.333	0.869
			Sometimes	37.0%		
			Usually	37.0%		
			Always	7.4%		
			(Missing)	0.0%		

* Ranking is based on the means.

** Responses are grouped as follows: Rarely/Never : 1, 2; Sometimes : 3; Usually : 4; Always : 5.

*** 54 Cases = 100%

Table 12 : Success of Software Development Projects

From Table 13, it can be seen that the use of a formal system development methodology had no significant impact on the success measures of software projects as all the respondents, whether or not their organisations adopted a formal system development methodology, indicated that their organisations achieved these measures quite frequently. This result may be supported by the observation in Fletcher and Hunt (1993) that the benefits of Software Engineering practices are typically long-term and difficult to measure. It is difficult for organisations to isolate the effects of Software Engineering practices on their software development work unless they undertake long-term, on-going studies with appropriate measurement techniques and tools.

Current Practice	**chi-sq.**	**Degree of Freedom**	**Sig.**	**Result***
Organisation uses formal system development methodology.	0.58	1	0.446	Not sig.
Organisation uses CASE tools.	4.18	1	0.041	Y>N
Current system development methods incorporate steps to improve maintainability of systems in future.	1.11	1	0.292	Not sig.

*Y - organisations which provided formal training in software engineering methods

N - organisations which did not provide formal training in software engineering methods

Table 13 : Impact on the Success Measures of Software Projects

3.6 Perception of Current Software Engineering Practices In Organisation

Data on the perception of current Software Engineering practices in the organisations which responded is shown in Tables 14 and 15.

Software Engineering Practices In The Organisation

Based on the mean scores of the responses, it was found that the respondents agreed in general to all the observations offered on the effects of Software Engineering practices on their organisations. The observations, ranked in order of decreasing mean scores obtained from the responses, were as follows:

- Software Engineering practices enable better control over software development and maintenance
- Software Engineering practices have improved the quality of systems developed/maintained
- Software Engineering practices improve the maintainability/adaptability of software
- Software Engineering practices improve the productivity of software developers/maintainers
- Software Engineering practices reduce the time to develop/maintain software
- Software Engineering practices have helped the organisation improve its competitive advantage
- Standard methodology adopted by the organisation is easy to use
- Object-oriented Software Engineering methods is/will be useful to your organisation
- Software Engineering practices help to reduce/contain IS cost.

These results support the literature in Fletcher and Hunt (1993) on the benefits which an organisation can expect to realise with the implementation of Software Engineering practices.

	Rank*	**Description**	**Response****	**Percent*****	**Mean**	**S.D.**

Perception	1	SE practices enable better	Agree	79.7%	4.020	0.520
Of SE		control over software	Neutral	11.1%		
Practices		development and	Disagree	0.0%		
In The		maintenance	(Missing)	9.3%		
Org'n	2	SE practices have	Agree	81.5%	4.000	0.456
		improved the quality of	Neutral	9.3%		
		systems developed/	Disagree	0.0%		
		maintained	(Missing)	9.3%		
	3	SE practices improve	Agree	81.5%	3.959	0.406
		maintainability/	Neutral	9.3%		
		adaptability of software	Disagree	0.0%		
			(Missing)	9.3%		
	4	SE practices improve the	Agree	72.2%	3.837	0.472
		productivity of software	Neutral	18.5%		
		developers/maintainers	Disagree	0.0%		
			(Missing)	9.3%		
	5	SE practices reduce the	Agree	63.0%	3.714	0.645
		time to develop/maintain	Neutral	24.1%		
		software	Disagree	3.7%		
			(Missing)	9.3%		
	5	SE practices have helped	Agree	61.2%	3.714	0.736
		the organisation improve	Neutral	24.1%		
		its competitive advantage	Disagree	5.6%		
			(Missing)	9.3%		
	6	Standard methodology	Agree	53.7%	3.604	0.707
		adopted by the organisation is	Neutral	29.6%		
		easy to use	Disagree	5.6%		
			(Missing)	11.1%		
	7	Object-oriented SE	Agree	48.2%	3.531	0.710
		methods is/will be	Neutral	37.0%		
		beneficial to your organisation	Disagree	5.6%		
			(Missing)	9.3%		
	8	SE practices help to	Agree	48.1%	3.490	0.711
		reduce/contain IS costs	Neutral	35.2%		
			Disagree	7.4%		
			(Missing)	9.3%		

* Ranking is based on the means.

** Responses are grouped as follows: Disagree : 1, 2; Neutral : 3; Agree : 4, 5.

*** 54 Cases = 100%

Table 14 : Perception of Current Software Engineering Practices

	Rank*	Description	Response**	Percent***	Mean	S.D.
Perception Of Barriers To Successful Adoption Of SE Practices	1	Lack of experienced IS	Agree	72.3%	3.759	0.799
		staff who can effectively	Neutral	18.5%		
		use SE methods/tools	Disagree	9.3%		
			(Missing)	0.0%		
	2	Lack of formal training in	Agree	70.4%	3.642	0.736
		SE engineering methods/	Neutral	16.7%		
		tools	Disagree	11.1%		
			(Missing)	1.9%		
	3	Lack of suitable	Agree	61.1%	3.593	0.765
		environment and tools to	Neutral	29.6%		
		support SE methods	Disagree	9.3%		
			(Missing)	0.0%		
	4	Legacy systems which are	Agree	55.6%	3.574	0.860
		not compatible with new	Neutral	33.3%		
		systems developed using	Disagree	11.1%		
		SE methods	(Missing)	0.0%		
	5	Complexity of SE	Agree	59.3%	3.528	0.749
		practices	Neutral	27.8%		
			Disagree	11.1%		
			(Missing)	1.9%		
	6	Lack of management	Agree	63.0%	3.500	0.947
		support	Neutral	20.4%		
			Disagree	16.7%		
			(Missing)	0.0%		
	7	Long learning curve	Agree	48.2%	3.531	0.710
			Neutral	37.0%		
			Disagree	5.6%		
			(Missing)	9.3%		
	8	Too large a financial	Agree	51.8%	3.444	0.925
		investment	Neutral	29.6%		
			Disagree	18.5%		
			(Missing)	0.0%		
	9	No clear objectives for	Agree	51.8%	3.377	0.860
		adopting SE practices	Neutral	29.6%		
			Disagree	16.7%		
			(Missing)	1.9%		
	10	Uncertainty over benefits	Agree	51.8%	3.264	0.944
		of adopting SE practices	Neutral	16.7%		
			Disagree	29.6%		
			(Missing)	1.9%		
	11	Unsuccessful previous	Agree	27.8%	3.077	0.710
		attempts in introducing	Neutral	48.1%		
		SE practices	Disagree	20.4%		
			(Missing)	3.7%		
	12	Standard methodology is	Agree	16.7%	2.759	0.799
		unsuitable for current	Neutral	42.6%		
		projects	Disagree	40.8%		
			(Missing)	0.0%		

* Ranking is based on the means. ** Responses are grouped as follows: Disagree : 1, 2; Neutral : 3; Agree : 4, 5.

*** 54 Cases = 100%

Table 15 : Perception of Barriers to Successful Adoption of Software Engineering Practices

Barriers To Successful Adoption Of Software Engineering Practices

Based on the mean scores of the responses, it was found that the respondents agreed in general to all, except one, of the factors cited as possible barriers to the successful adoption of Software Engineering practices in their organisations. The factor to which the respondents took exception was “Standard methodology is unsuitable for current projects”. This is an encouraging result, but it cannot be guaranteed in the long term if an organisation based its choice of Software Engineering practices solely on current projects. Fletcher and Hunt (1993) reminded that there is a need to “gaze into the crystal ball” when making decisions on Software Engineering practices to adopt i.e. organisations need to be visionary and determine how they will use the technology two to five years from now in order for such a result to remain.

The rest of the factors to which the respondents generally agreed, ranked in order of decreasing mean scores obtained from the responses, were as follows:

- Lack of experienced IS staff who can effectively use Software Engineering methods/tools
- Lack of formal training in Software Engineering methods/tools
- Lack of suitable environment and tools to support Software Engineering methods
- Legacy systems which are not compatible with new systems developed using Software Engineering methods
- Complexity of Software Engineering practices
- Lack of management support
- Long learning curve
- Too large a financial investment
- No clear objectives for adopting Software Engineering practices
- Uncertainty over benefits of adopting Software Engineering practices
- Unsuccessful previous attempts in introducing Software Engineering practices.

4.0 Hypotheses Analysis

The results of this study are analysed and used to answer the research hypotheses stated in section 2.1.

Objective (a):
To find out the extent to which Software Engineering practices are carried out in Singapore.

<u>Hypothesis 1</u>

Software Engineering is practised in the majority of organisations in Singapore.
True. 68.5% of organisations in Singapore used a formal system development methodology.

<u>Hypothesis 2</u>

Software Engineering techniques are more widely used than Software Engineering tools (due to historical and cost reasons).

True. Only 29.6% of organisations used CASE tools whereas over 70% of organisations used at least one Software Engineering technique during the requirements analysis, system design, review/testing and maintenance phases. The results of the survey reported in (Glass, 1996) also indicate a low usage of CASE tools by organisations in the United States.

Objective (b):

To identify the most commonly used software process models and techniques.

<u>**Hypothesis 3**</u>

The sequential/waterfall model is the most commonly used software process model.

True. The most commonly used software process model was the sequential/waterfall model (37%), followed by the incremental model (16.7%), the rapid prototyping model (9.3%), the spiral model (3.7%) and others (1.9%). 11.1% of respondents were not sure which model they were using.

<u>**Hypothesis 4**</u>

The percentage of organisations which are currently using object-oriented techniques is relatively low.

True. The study showed that 9.3% of organisations were using object-oriented analysis techniques and 3.7% of organisations were using object-oriented design techniques. 10% of organisations said the measures taken to improve future maintainability of systems included object-oriented approaches. 48.2% of respondents agreed that object-oriented Software Engineering methods were/would be useful to their organisation.

<u>**Hypothesis 5**</u>

Software Engineering techniques and tools are used more extensively in the analysis and design phases of the software production process.

True. On average, an organisation used 4 different Software Engineering techniques in the analysis phase, 3 in the design phase and 1 each in the coding, review/testing and maintenance phases.

<u>**Hypothesis 6**</u>

Some techniques are more commonly used than others.

True. The 4 requirements analysis techniques used by more than half of the respondents were data flow diagrams (77.8%), data dictionary (57.4%), entity relationship diagrams (51.9%) and flowcharts (51.9%). The 3 most commonly used system design techniques were screen design

(79.6%), report design (68.5%) and structure charts (40.7%). For coding, program specifications and 4GL had the highest usage at 61.1% and 63% respectively. The most widely used review technique was walkthroughs at 72.2%. The most common Software Engineering practice instituted for the maintenance phase was software change management procedures (72.2%), followed by documentation change management procedures (57.4%).

Objective (c):
To identify factors which influence software engineering practices.

Hypothesis 7
The adoption of Software Engineering practices is influenced by:
a. size of organisation
b. industry sector/type
(a) - Not True. The results showed that a higher percentage of small organisations were using a formal system development methodology compared to large companies. However, it was found that organisations with more IT professionals were more likely to be using a formal system development methodology than organisations with fewer IT professionals.
(b) - Partially True. In terms of industry sector, the results showed that the public sector led in the use of formal system development methodology while the private sector led in the use of CASE tools. In terms of industry type, the results showed that there was no significant difference between different types of industry in the adoption of Software Engineering practices.

Hypothesis 8
Formal training has a positive effect on the adoption of Software Engineering practices.
True. It was found that organisations which provided formal training in Software Engineering methods were more likely to use CASE tools than organisations which did not provide formal training in Software Engineering. However, there was no significant relationship found between the provision of formal training and the adoption of a formal system development methodology.

Hypothesis 9
Organisations where the IT function has a high profile tend to implement Software Engineering practices more extensively.
True. It was found that IS functions which operated as an independent unit in the organisation, reporting directly to the chief executive officer, were more likely to be using a formal system development methodology than IS functions which were totally or partially subsumed under another functional unit in the organisation.

<u>Hypothesis 10</u>

The adoption of a system development methodology forms the basis for use of Software Engineering techniques and tools.

Partially True. It was found that organisations which had adopted a formal system development methodology were definitely more likely to use Software Engineering techniques in all stages of the software life cycle than organisations which had not adopt a formal methodology. However, No correlation was found between the use of a formal system development method and the use of CASE tools.

Objective (d):

To identify the problems organisations face in adopting Software Engineering practices.

<u>Hypothesis 11</u>

There are common problems faced by organisations in adopting Software Engineering practices.

True. The problems identified, ranked in order of decreasing significance, were as follows:

- Lack of experienced IS staff who can effectively use Software Engineering methods/tools
- Lack of formal training in Software Engineering methods/tools
- Lack of suitable environment and tools to support Software Engineering methods
- Legacy systems which are not compatible with new systems developed using Software Engineering methods
- Complexity of Software Engineering practices
- Lack of management support
- Long learning curve
- Too large a financial investment
- No clear objectives for adopting Software Engineering practices
- Uncertainty over benefits of adopting Software Engineering practices
- Unsuccessful previous attempts in introducing Software Engineering practices.

Objective (e):

To gain an insight into organisations' perception towards Software Engineering practices.

<u>Hypothesis 12</u>

In general, organisations have realised benefits with the implementation of Software Engineering practices.

True. Organisations had the following perceptions, ranked in order of decreasing significance, towards the Software Engineering practices in their organisations:

- Software Engineering practices enable better control over software development and maintenance
- Software Engineering practices have improved the quality of systems developed/maintained
- Software Engineering practices improve the maintainability/adaptability of software
- Software Engineering practices improve the productivity of software developers/maintainers
- Software Engineering practices reduce the time to develop/maintain software
- Software Engineering practices have helped the organisation improve its competitive advantage
- Standard methodology adopted by the organisation is easy to use
- Object-oriented Software Engineering methods is/will be useful to the organisation
- Software Engineering practices help to reduce/contain IS cost.

<u>Hypothesis 13</u>

Software Engineering practices have a positive effect on measures of software development success.

Not True. The study showed that the use of a formal system development methodology had no significant impact on the success measures of software projects as all the respondents, whether or not their organisations adopted a formal system development methodology, indicated that their organisations achieved these measures reasonably frequently.

5.0 Implications of Study

In general, the majority of organisations which develop software systems made use of Software Engineering techniques. However, the usage of Software Engineering techniques was still concentrated in the analysis and design phases of the software life cycle. In the coding, review/testing and maintenance phases, the use of Software Engineering techniques was very limited - each organisation on average used only one technique in each of these phases. The level of usage of CASE tools was also fairly low. Organisations in Singapore should increase the application of Software Engineering techniques and tools so as to alleviate the long-standing problems associated with software development and maintenance.

Although the implementation of Software Engineering techniques and tools needs to be improved upon, respondents' perception of Software Engineering issues were found to support the work of researchers and practitioners with regards to the benefits of Software Engineering practices and the barriers to successful adoption of these practices. This shows that organisations are generally well aware of the benefits and problems in using Software Engineering techniques and tools.

Major barriers to the implementation of Software Engineering practices were the lack of experienced staff who can effectively use Software Engineering techniques and tools and the lack of formal training in these areas. Companies need to be willing to invest in training for their staff and provide a conducive environment for staff to gain experience in the use of Software Engineering techniques and tools. This is important as "experience" is perceived as the most important hiring criteria for a position in an IS organisation in the United States (Glass, 1996).

6.0 Conclusion

In this paper we discussed the results and implications of a survey on Software Engineering practices in Singapore conducted in 1995. The study was intended to gain an insight into the extent to which Software Engineering practices have been adopted by organisations in Singapore, the benefits these organisations have realised, the problems they face in implementing these practices and their perceptions on issues prevalent in the Software Engineering field.

The study indicated that there is much for Singapore organisations to do in adopting state-of-the-art practices in the Software Engineering field. However, our Software Engineering practices are comparable to our counterparts in the United States and Hong Kong.

References

Conger, S A, (1994), *The New Software Engineering*, Wadsworth Publishing.

Degrace, P, and Stahl, L H, (1990), *Wicked Problems, Righteous Solutions: A Catalogue of Modern Software Engineering Paradigms*, Yourdon Press.

Fletcher, T, and Hunt, J, (1993), *Software Engineering and CASE: Bridging The Culture Gap*, McGraw Hill.

Glass, R.L., (1996), *Results of the First IS State-of-the-Practice Survey,* The Software Practitioner, May-June-July-August.

Keyes, J, (1992), *Software Engineering Productivity Handbook*, McGraw Hill.

Khor, C K, and Woo, L Y, (1993/94), *Software Project Management Practices In Singapore*, Department Of Information Systems and Computer Science, National University Of Singapore, 1993/94.

Mandrioli, D, and Meyer, B, (1992), *Advances in Object-Oriented Software Engineering*, Prentice Hall.

Phoon, W T, and Poo, C C, (1990/91), Use of Software Development Methods In Singapore, Department Of Information Systems and Computer Science, National University Of Singapore, 1990/91.

Schach, S R, (1993), *Software Engineering*, 2nd Ed., Richard D. Irwin Inc and Aksen Associates Inc.

Yip, S W L, Lam, T, and Chan, S K M, A Software Maintenance Survey, *IEEE* 0-8186-6960-8/94, 1994, pp 70-79.