CASE and Software Maintenance 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

Many claims have been made about the benefits of CASE and many problems have been reported about software maintenance. This paper discusses two aspects from a survey recently conducted in Singapore on Software Engineering Practices. These two aspects include : how organisations in Singapore use CASE tools and how organisations in Singapore practise software maintenance.

The survey indicated that the level of usage of CASE tools in Singapore is fairly low. Organizations in Singapore are generally well aware of the benefits and problems in CASE. The major barriers to the use of CASE tools were the high cost of implementing CASE tools and the long learning curve to use CASE tools effectively. On the practice of software maintenance, the survey found that the greatest problem with it was staff turnover.

1. Introduction

A survey to ascertain the extent of Software Engineering practices in Singapore was conducted in 1995. The aim of the survey was to gather an understanding of how organisations in Singapore have adopted Software Engineering practice in the development of information systems so that effort can be channelled to address areas where improvements can be made.

The survey was guided by 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.

In the context of the study, "Software Engineering Practices" include methodologies, models, techniques and tools used in the Software Engineering process. Fletcher & Hunt [2] defines methodologies, models, techniques and tools in the following manner :

- 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.

As the topics covered in the survey are wide-ranging, this paper shall discuss the following issues :

1. How organisations in Singapore use CASE tools.
2. How organisations in Singapore practise software maintenance.

Other topics as covered in the survey have been reported and documented in [3].

The usual definition of CASE tools only includes tools that support the planning, analysis, design and construction phases. In this study, CASE tools will also include those tools supporting project management, configuration management, testing and reverse engineering.

The nature and sophistication of CASE tools and software maintenance practices were measured by the following variables:

- CASE Tools
  Existence and extent of usage of CASE tools; and perception of benefits and barriers relating to the use of CASE tools.
- Software Maintenance
  Software maintenance activities and practices.

This paper is organised in the following manner :

Section 2 : Discusses the set of hypotheses formulated for the purpose of the study.
Section 3 : Summarises the demographic characteristics of respondents.
Section 4 : Discusses how organisations in Singapore currently practise CASE.
Section 5 : Discusses the perceptions of organisations on the use of CASE tools in their Software Engineering process. Both the benefits and barriers to using CASE tools will be discussed.
Section 6 : Sub-group analysis is carried out in the study and this section reports on the results.
Section 7 : Summarises the findings on the use of CASE tools by validating the hypotheses listed in Section 2.
Section 8 : Discusses how organisations in Singapore currently practise software maintenance.
Section 9 : Summarises the findings on software maintenance practices in Singapore by validating the hypotheses listed in Section 2.
Section 10 : Concludes the discussion.

2. Hypotheses

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

Hypothesis 1:

There are common problems faced by organisations in using CASE tools.

Rationale: Many researchers have documented the problems faced by organisations in using CASE tools. This hypothesis is to test whether these problems are also faced by the organisations surveyed.

Hypothesis 2:

Organisations have realised benefits with the use of CASE tools.

Rationale: Many researchers have documented the benefits which should be realised by organisations which use CASE tools. This hypothesis is to test the perception of organisations surveyed towards the realisation of these benefits.

Hypothesis 3:

In general, certain types of software maintenance will be carried out more often than others.

Rationale: Software maintenance can be classified into various types of activities e.g. error correction, enhancements etc. This hypothesis is to find out which types of software maintenance are carried out more often in the organisations surveyed.

Hypothesis 4:

In general, most of the time spent on software maintenance will be in designing, coding and testing the changes.

Rationale: This hypothesis is to test whether it is correct to assume that the designing, coding and testing of changes take up the most time in maintenance activities.

Hypothesis 5:

The major problem in software maintenance is the lack of software engineering discipline.

Rationale: Software engineering is said to improve the software maintenance process. This hypothesis is to test whether the lack of software engineering discipline is actually a major problem in software maintenance.

3. Demographic Characteristics of Respondents

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.

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, thus lending strength to the validity of the responses.

The manufacturing industry dominated the percentage of respondents at 24.1%, followed by the IT industry at 18.5%, the banking/finance/insurance industry at 14.8% and the

retail/trade/tourism industry at 11.1%. The remaining 31.6% of the respondents were from the transportation, utility/telecommunications, education/training and other types of industries.

IT users 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.

Microcomputers and minicomputers were the most widely used types of computers at 81.5% and 72.2% respectively. While the percentage of mainframe users was the lowest at 31.5%.

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(Information Systems) 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.

The majority (53.7%) of IS departments functioned as independent units reporting directly to the Chief Executive Officer of the organisation while 33.3% of the IS departments were subsumed under another functional unit in the organisation.

The majority of the organisations were working towards ISO 9000 certification (40.7%), 27.8% had attained the ISO 9000 certification and an equal percentage had no intention of going for certification at the point in time.

4.0 Current Practices on Use of CASE Tools

Table 4.1 shows that only 29.6% of organisations which responded used CASE tools although 68.5% of these organisations used a formal software development methodology.

	Response	Freq	Percent
Organisation Uses CASE Tools	Yes	16	29.6%
	No	36	66.7%
	(Missing)	2	3.7%
	Total	54	100.0%

Table 4.1 : Organisation Using CASE Tools

The majority (43.8%) of organisations which had CASE tools used them in a quarter or less of their software systems under development (see Table 4.2). 31.3% of them used the CASE tool for between a quarter to half of their systems under development. Only 18.8% of them used their CASE tool for more than 75% of their applications under development.

% of Software Systems Currently Developed Using CASE Tools	Response	Freq	Percent*
	0 -25%	7	43.8%
	26 - 50%	5	31.3%
	51 - 75%	1	6.3%
	76 - 100%	3	18.8%
	Total	16	100.0%

*16 Cases = 100% (based on number of respondents who used CASE tools as shown in Table 4.1)

Table 4.2 : Development Using CASE Tools

% of Software Systems Currently Maintained Using CASE Tools	Response	Freq	Percent*
	0 -25%	10	62.5%
	26 - 50%	3	18.8%
	51 - 75%	1	6.3%
	76 - 100%	2	12.5%
	Total	16	100.0%

*16 Cases = 100% (based on number of respondents who used CASE tools as shown in Table 4.1)

Table 4.3 : Software Maintained Using CASE Tools

According to Table 4.3, the usage of CASE tools for maintenance of software systems was even lower. 62.5% of organisations with CASE tool used it to maintain up to only a quarter of their systems under maintenance. Only 12.5% of those with CASE tools used it to maintain more than 75% of their systems under maintenance.

In a case study conducted by Mary Sumner of Southern Illinois University [1] on 13 project managers who used CASE tools, it was found that approximately 40% of new applications were developed with CASE tool support while only 21% of maintenance projects had CASE tool support. Hence, the observed trend is that CASE tools are used more for software development than software maintenance, regardless of geographical location.

Table 4.4 shows that the four most common activities for which CASE tools were used were:

- Documentation (62.5%)
- System design (56.3%)
- Requirements analysis (50%)
- Drawing diagrams (50%)

The other more common activities which involve the use of CASE tools were information systems planning (37.5%), coding or code generation (37.5%), prototyping (31.3%), project management (25%) and change management (25%).

The remaining activities which involved the use of CASE tools were testing (18.9%), configuration management (12.5%) and others (6.3%).

In the study conducted by Sumner, it was also observed that CASE tools were used to support isolated activities within the system development life cycle, rather than being integrated throughout the life cycle. In Sumner's study, similar to the current study, the three activities for which CASE tools were most frequently used were system analysis, system design and documentation, showing that there is a similar pattern in the usage of CASE tools regardless of geographical location.

	Rank	Response	Freq	Percent*
Activities For Which CASE Tools Are Used	1	Documentation	10	62.5%
	2	System Design	9	56.3%
	3	Requirements Analysis	8	50.0%
	3	Drawing Diagrams	8	50.0%
	4	Information Systems Planning	6	37.5%
	4	Coding	6	37.5%
	5	Prototyping	5	31.3%
	6	Project Management	4	25.0%
	6	Change Management	4	25.0%
	7	Testing	3	18.9%
	8	Configuration Management	2	12.5%
	9	Others	1	6.3%

*16 Cases = 100% (based on number of respondents who used CASE tools as shown in Table 4.1)

Table 4.4 : Activities For Which CASE Tools Are Used

5.0 Perception on the Use of CASE Tools

Respondents were asked to indicate their perception of the benefits and barriers with regards to the use of CASE tools in the survey. This section discusses the results gathered from the survey.

5.1 Perception of Benefits of CASE Tools

Many claims have been made about the benefits of CASE. One of the major benefits of CASE is the introduction of engineering-like discipline into the software process. Using CASE tools, a software engineer can take advantage of diagramming tools, design checking tools and a disciplined methodology [4]. This sub-section examines the perception of benefits of CASE tools by the organisations in Singapore.

From the respondent's perception as shown in Table 5.1, the four *most significant* benefits realized with the use of CASE tools were:

- Facilitates drawing of diagrams (62.5%).
- Improves software maintenance (56.3%).
- Creates a repository for system documentation (56.3%).
- Provides checks on analysis and design errors (56.3%).

	Rank	Response	Freq	Percent*
Benefits Realized With Use Of CASE Tools	1	Facilitates drawing of diagrams	10	62.5%
	2	Improves software maintenance	9	56.3%
	2	Creates repository for system documentation	9	56.3%
	2	Provides checks on analysis and design errors	9	56.3%
	3	Increases user involvement in system design	6	37.5%
	3	Creates enterprise-wide data dictionary	6	37.5%
	3	Aids project management and control	6	37.5%
	4	Support for software engineering methods	5	31.3%
	4	Ensures conformance to system development and maintenance standards	5	31.3%
	5	Supports prototyping	4	25.0%
	6	Others	1	6.3%

*16 Cases = 100% (based on number of respondents who used CASE tools as shown in Table 4.1)

Table 5.1 : Benefits Realized With Use of CASE Tools

The other *more significant* benefits perceived by the respondents, each with a response of 37.5%, were:

- Increases user involvement in system design.
- Creates an enterprise-wide data dictionary.
- Aids in project management and control.

The *less significant* benefits from using CASE tools are perceived to be:

- Support for software engineering methods (31.3%).
- Ensures conformance to system development and maintenance standards (31.3%).
- Supports prototyping (25%).
- Others (6.3%).

In the study conducted by Sumner, the five most significant benefits of using CASE tools, as perceived by the respondents, were:

- Vehicle for using structured design.
- Prevents re-drawing of diagrams.
- Provides improved maintenance.
- Increases user involvement in system design.
- Creates a repository for design documentation.

This shows that CASE tools are perceived to be quite useful as a diagramming tool, a repository for system documentation and a means to improve software maintenance by the respondents in both USA and Singapore.

5.2 Perception of Barriers to Use of CASE Tools

Based on the mean scores of the responses, it was found that the respondents agreed in general that the following factors, ranked in order of decreasing mean scores, were barriers to the use of CASE tools (see Table 5.2):

- High cost of implementing CASE tools.
- Long learning curve to use CASE tools effectively.
- Limited capability of CASE tools.
- Lack of fit between system development methodology and CASE tools.
- Using CASE tools without knowledge of underlying software engineering methods and techniques.
- Uncertainty over the benefits of CASE tools.

The respondents in general disagreed that resistance by system developers was a barrier to the use of CASE tools.

In the study conducted by Sumner, both CASE and non-CASE users felt that the limited capability of CASE tools was the most significant barrier to the use of CASE tools. For CASE users, other significant barriers were the long learning curve and lack of fit with current methodology. For non-CASE users, the other very significant barrier was the high cost which, not surprisingly, was the least of the CASE users' concerns. Resistance by system developers was ranked last and second last by non-CASE and CASE users respectively.

The results of the current study are comparable to those obtained from Sumner's study. As the current study had a high proportion of non-CASE users, the high cost of CASE tools was ranked as the most significant barrier; the long learning curve was the next most significant barrier followed by the limited capability of CASE tools; resistance by system developers was

also given the lowest ranking in the current study, showing that IT professionals had a similar perception towards CASE tools in both the US and Singapore.

6.0 Sub-Group Analysis

This section discusses the results from sub-group comparisons.

	Rank*	Factors	Response**	Percent***	Mean	S.D.
Factors Which Are Barriers To Use Of CASE Tools	1	High cost of implementing	Agree	46.3%	3.784	0.917
		CASE tools	Neutral	14.8%		
			Disagree	7.4%		
			(Missing)	31.5%		
	2	Long learning curve to use	Agree	48.1%	3.676	0.852
		CASE tools effectively	Neutral	11.1%		
			Disagree	9.3%		
			(Missing)	31.5%		
	3	Limited capability of	Agree	35.2%	3.595	0.927
		CASE tools	Neutral	25.9%		
			Disagree	7.4%		
			(Missing)	31.5%		
	4	Lack of fit between system	Agree	38.9%	3.526	0.922
		development methodology	Neutral	20.4%		
		and CASE tools	Disagree	11.1%		
			(Missing)	29.6%		
	5	Using CASE tools without	Agree	35.2%	3.472	1.055
		knowledge of underlying	Neutral	18.5%		
		software engineering	Disagree	13.0%		
		methods & techniques	(Missing)	33.3%		
	6	Uncertainty over benefits	Agree	31.5%	3.135	1.058
		of using CASE tools	Neutral	9.3%		
			Disagree	27.8%		
			(Missing)	31.5%		
	7	Resistance by system	Agree	22.2%	2.946	0.970
		developers	Neutral	16.7%		
			Disagree	29.6%		
			(Missing)	31.5%		

* Ranking is based on the means.

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

*** 54 Cases = 100%

Table 5.2 : Barriers to Use of CASE Tools

6.1 Differences By Demographic Characteristics

A series of tests were conducted to find out whether there were significant differences in the practices between organisations with different demographic characteristics.

Number Of IT Professionals In Organisation

It was found that organisations with 20 or more IT professionals had a higher tendency to adopt a formal system development methodology and use CASE tools than organisations with less than 20 IT professionals (chi-square=7.02, d.f.(degree of freedom)=1, significance level=0.008). This supports the assumption that larger IT departments would take the lead in adopting software engineering methods and tools since a larger IT department is an indication of the size and complexity of projects undertaken and the importance of the IT function to the organisation.

Number Of Employees In Organisation and Industry Type

Intuitively, the size and nature (industry type) of the organisation would influence the number of IT employees in the organisation and its software engineering practices. However, a higher percentage of smaller organisations (i.e. those with less than 500 employees) made use of a formal system development methodology and CASE tools compared with larger organisations(31.0% vs 28.0%). In addition, there is no significant difference between the various types of industry in terms of number of IT professionals employed (chi-square=12.29, d.f.=7, sig=0.091), the use of a formal system development methodology (chi-square=13.11, d.f.=7, sig=0.069) and the use of CASE tools (chi-square=8.95, d.f.=7, sig=0.256).

A possible explanation for this could be that the size and nature of the organisation does not indicate the importance of IT to the organisation as much as the number of IT professionals in the organisation, leading to the conclusion that the use of software engineering methods and tools is influenced more by the importance of IT to the organisation rather than the physical size or nature (industry type) of the company.

Industry Sector

Interestingly, the results from the survey show that although the public sector led in the usage of formal system development methodology (78.6% vs 65.0%), the private sector led in the usage of CASE tools (30.0% vs 28.6%).

The higher usage of formal system development methodology in public sector organisations is expected as the computerization of the majority of public sector organisations come under the Civil Service Computerization Scheme managed by the National Computer Board. The National Computer Board's policy is that all its computerization projects must follow its formal methodology - Application Development and Maintenance Methodology. In the private sector, the organisations may not have the manpower or financial resources to implement a formal system development methodology. Even if they have the necessary resources, the management's policy may not be to enforce the use of a software engineering methodology.

With regards to the use of CASE tools, the public sector may be lagging slightly behind the private sector, because public sector organisations usually need to justify the purchase of CASE tools with concrete benefits. Respondents from organisations which did not use CASE tools felt that the uncertainty over the benefits of using CASE tools and the high cost of implementing CASE tools were significant barriers to the use of CASE tools. Since the benefits of using CASE tools are not easy to quantify, it may have been difficult for public sector organisations to justify the purchase of CASE tools. In the private sector, the purchase of CASE tools may not be governed so strictly by the results of a cost-benefit analysis, thus leading to more wide-spread use of CASE tools.

Provision Of Formal Training In Software Engineering Methods

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 (chi-square=4.18, d.f.=1, sig.=0.041). In addition, organisations which provided formal software engineering training were more likely to have engaged external consultants to assist in the implementation of software engineering methods and tools (chi-square=8.57, d.f.=1, sig=0.003). These findings support the observation made in Sumner's paper [1] that "CASE tools do nothing unless you understand and apply the underlying principles of software engineering". Besides choosing the right software development methodology as a foundation, organisations also need to provide significant training and to modify the IS culture to properly support the new software engineering practices so as to maximize their benefits gained from using CASE tools.

6.2 Differences By Current Practices

Tests were conducted to find out whether there were significant differences in terms of current practices and perception between organisations which used a formal system development methodology and CASE tools and those which did not.

As shown in Table 6.1, there were only two significant differences in the perceptions toward barriers to the use of CASE tools between respondents from organisations which used CASE tools and those which did not. These two differences were:

i. Respondents from organisations which did not use CASE tools were more likely to agree that uncertainty over benefits of using CASE tools was a barrier to the use of CASE tools than those from organisations which used CASE tools (chi-square=5.74, d.f.=1, sig.=0.017).
ii. Respondents from organisations which did not use CASE tools were also more likely to agree that the high cost of implementing CASE tools was a barrier to the use of CASE

tools than those from organisations which used CASE tools (chi-square=10.89, d.f.=1, sig=0.001).

Use Of Case Tools

Barriers To The Use Of CASE Tools	chi-sq.	D.F.	Sig.	Result*
Limited capability of CASE tools	0.25	1	0.616	Not sig.
Long learning curve to use CASE tools effectively	0.13	1	0.716	Not sig.
Lack of fit between system development methodology and CASE tools	0.28	1	0.594	Not sig.
Uncertainty over benefits of using CASE tools	5.74	1	0.017	N>Y
Resistance by system developers	0.72	1	0.395	Not sig.
High cost of implementing CASE tools	10.89	1	0.001	N>Y
Using CASE tools without knowledge of underlying software engineering methods & techniques	0.11	1	0.095	Not sig.

*N - Organisations which did not use CASE tools

Y - Organisations which used CASE tools

Table 6.1 : Significant Differences in Perceptions Toward Barriers to the Use of CASE Tools

These results could be a reflection of the justification policy for IS projects practiced by many organisations. According to Sumner [1], IS projects have traditionally been justified using cost-benefit analysis to demonstrate the return on investment. However, a cost-benefit strategy limits the projects that can be justified to those whose outcomes are clear. These projects are typically operational-type projects which cut costs by automating certain business functions. In contrast, projects which are more innovative and carry potential risk are more difficult to justify using cost-benefit techniques. As such an investment in CASE tools may not be easy to justify using a return on investment approach if the organisation is uncertain of the benefits of using CASE tools and the cost of implementing the CASE tools is high. If an organisation is willing to pursue innovative technology projects that are driven by a business justification, rather than by expected financial returns, it will be more likely to adopt the use of CASE tools.

Use Of Formal System Development Methodology & Use Of CASE Tools

No correlation was found between the use of a formal system development method and the use of CASE tools.

7.0 Summary of Findings on the Use of CASE Tools

The results are summarized in the form of answers to the research hypotheses stated in section 2.

Hypothesis 1

There are common problems faced by organisations in using CASE tools.

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

- High cost of implementing CASE tools.
- Long learning curve to use CASE tools effectively.
- Limited capability of CASE tools.
- Lack of fit between system development methodology and CASE tools.
- Using CASE tools without knowledge of underlying software engineering methods and techniques.
- Uncertainty over the benefits of CASE tools.

Hypothesis 2

Organisations have realised benefits with the use of CASE tools.

True. From the respondent's perception, the four most significant benefits realised with the use of CASE tools were:

- Facilitates drawing of diagrams (62.5%).
- Improves software maintenance (56.3%).
- Creates a repository for system documentation (56.3%)
- Provides checks on analysis and design errors (56.3%).

8.0 Software Maintenance

The survey showed that the most significant types of software maintenance carried out were error correction and system enhancement, each being carried out by 88.9% of the respondents' organisations (see Table 8.1).

	Rank	Response	Freq	Percent*
Activities Carried Out During Software Maintenance	1	Error correction	48	88.9%
	1	Enhancements	48	88.9%
	2	Performance tuning	34	63.0%
	3	Adapting to new hardware/software/work environment	33	61.1%
	4	Answering questions	27	50.0%
	5	Documentation	26	48.1%
	6	Re-engineering, renewal, retrofitting	12	22.2%
	7	Others	1	1.9%

*54 Cases = 100%

Table 8.1 : Activities Carried Out During Software Maintenance

The other more significant types of software maintenance carried out were performance tuning (63%), adapting to new hardware/software/work environment (61.1%), answering questions from users (50%) and documentation (48.1%). Maintenance arising from re-engineering, renewal and retrofitting was relatively low (22.2%), this may be explained by the fact that re-engineering, renewal and retrofitting is not carried out by many organisations in Singapore, although the trend may be increasing. These results are comparable to the findings from the

software maintenance study conducted in Hong Kong by Yip, Lam and Chan [5], where the most significant type of software maintenance was also enhancement, followed by, in decreasing order of significance, error correction, answering questions, adapting to new hardware/software/work environment, documentation, performance tuning and finally, re-engineering.

	Type of Activity	Response*	Freq	Percent**
Relative Amount of Time Spent On Different Maintenance Activities	Studying requests	6	13	**24.1%**
		5	7	13.0%
		4	9	16.7%
		3	12	22.2%
		2	7	13.0%
		1	5	9.3%
	Designing changes	6	3	5.6%
		5	15	**27.8%**
		4	14	25.9%
		3	11	20.4%
		2	5	9.3%
		1	3	5.6%
	Coding changes	6	6	11.1%
		5	11	20.4%
		4	15	**27.8%**
		3	12	22.2%
		2	6	11.1%
		1	1	1.9%
	Testing changes	6	7	13.0%
		5	10	18.5%
		4	7	13.0%
		3	13	**24.1%**
		2	12	22.2%
		1	4	7.4%
	Implementing changes (includes training, documentation, etc.)	6	8	14.8%
		5	5	9.3%
		4	5	9.3%
		3	9	16.7%
		2	19	**35.2%**
		1	7	13.0%
	Others	6	1	1.9%
		5	0	0.0%
		4	0	0.0%
		3	0	0.0%
		2	1	1.9%
		1	11	**20.4%**

* Ranking scale where 6 = most time spent and 1 = least time spent.

**54 Cases = 100%. Highlighted percentage shows the mode of responses for the type of activity.

Table 8.2 : Time Spent on Maintenance Activities

During maintenance, the largest amount of time was spent studying the requests, followed by, in order of decreasing amount of time spent, designing the changes, coding the changes, testing

the changes, implementing the changes and other activities (see Table 8.2). In comparison, the Hong Kong study found that the largest amount of time was spent in coding, followed by design, studying the requests, testing and implementation.

The greatest problem faced by organisations in software maintenance was staff turnover, ranked first by 61.1% of the respondents. This is shown in Table 8.3. The next most significant problem encountered was the lack of manpower resources (59.3%), followed by changing user requirements (57.4%), and lack of documentation (53.7%). Other fairly common problems encountered were problems with upgrades and data conversion (38.9%), different programming styles used (35.2%) and lack of tools to facilitate maintenance work. Relatively few of the respondents (25.9%) felt that the lack of software engineering discipline contributed to the difficulties of software maintenance. This may be due to a failure to recognize how software engineering discipline can eliminate some of the problems of software maintenance such as the lack of proper documentation which many had cited as a major problem.

	Rank	Response	Freq	Percent*
Problems Encountered With Software Maintenance	1	Staff turnover	33	61.1%
	2	Lack of manpower resources	32	59.3%
	3	Changing user requirements	31	57.4%
	4	Lack of documentation	29	53.7%
	5	Problems with upgrades and data conversion	21	38.9%
	6	Different programming styles used	19	35.2%
	7	Lack of tools to facilitate maintenance work	18	33.3%
	8	Lack of software engineering discipline	14	25.9%
	9	Others	1	1.9%

*54 Cases = 100%

Table 8.3 : Problems Encountered With Software Maintenance

In comparison, the study conducted in Hong Kong showed that the largest category of problems encountered in maintenance were environmental factors such as the lack of manpower resources, different programming styles used, insufficient documentation, problems with upgrades/data conversion and lack of suitable tools. The lack of appropriate management policy was ranked second in significance, the problem of staff turnover was ranked third, while changing user requirements was ranked fourth.

74.1% of the respondents replied that their organisation's current system development methods incorporate measures to improve the maintainability of systems in future. Out of these, 72.5% indicated that the measures taken included the institution of quality standards. The other measures taken by them to improve maintainability of systems included the use of CASE tools (20%), other methods (17.5%), use of code generators (15%) and use of object-oriented approaches (10%). In the study conducted in Hong Kong, only 8 out of 49 respondents used CASE tools (see Tables 8.4 and 8.5).

Does Current System Devt. Methods Incorporate Steps To Improve Maintain-ability Of Systems In Future?	Response	Freq	Percent
	Yes	40	74.1%
	No	13	24.1%
	(Missing)	1	1.9%
	Total	54	100.0%

Table 8.4 : Improving Software Maintainability

Measures Employed to Improve Maintainability Of Systems In Future	Rank	Response	Freq	Percent*
	1	Instituting quality standards	29	72.5%
	2	Use of CASE tools	8	20.0%
	3	Others	7	17.5%
	4	Use of application code generators	6	15.0%
	5	Use of object-oriented approaches	4	10.0%

*40 Cases = 100% (based on number of respondents who replied 'Yes' in Table 4.8)

Table 8.5 : Measures for Improving Software Maintainability

9.0 Summary of Findings on Software Maintenance Practices

The results are summarized in the form of answers to the research hypotheses stated in section 2.

Hypothesis 3

In general, certain types of software maintenance will be carried out more often than others.

True. The survey showed that the most significant types of software maintenance were error correction and system enhancement, each being carried out by 88.9% of the respondents' organisations. The other more significant types of software maintenance carried out were performance tuning (63%), adapting to new hardware/software/work environment (61.1%), answering questions from users (50%) and documentation (48.1%). Maintenance activities arising from re-engineering, renewal and retrofitting were relatively low (22.2%).

Hypothesis 4

In general, most of the time spent on software maintenance will be in designing, coding and testing the changes.

Partially True. It was found that during maintenance, the largest amount of time was spent studying the requests, followed by, in order of decreasing amount of time spent, designing the changes, coding the changes, testing the changes, implementing the changes and other activities.

Hypothesis 5

The major problem in software maintenance is the lack of software engineering discipline.

Not True. The study showed that the greatest problem faced by organisations in software maintenance was staff turnover, ranked first by 61.1% of the respondents. The next most significant problem encountered by the organisations was the lack of manpower resources (59.3%), followed by changing user requirements (57.4%), and lack of documentation (53.7%). Other fairly common problems encountered were problems with upgrades and data conversion (38.9%), different programming styles used (35.2%) and lack of tools to facilitate maintenance work. Relatively few of the respondents (25.9%) felt that the lack of a software engineering discipline contributed to the difficulties of software maintenance.

10.0 Conclusion

The level of usage of CASE tools in Singapore is fairly low. Organisations in Singapore are generally well aware of the benefits and problems in using software engineering techniques and tools. Major barriers to the use of Case tools were the high cost of implementing CASE tools and the long learning curve to use CASE tools effectively. The lack of experienced staff who can effectively use software engineering techniques and tools and the lack of formal training in these areas have also compounded the problems. 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.

The greatest problem with software maintenance was staff turnover. Organisations need to intensify their efforts to institute measures which will reduce the impact of staff turnover on software maintenance. These measures include the use of a formal system development methodology, CASE tools and object-oriented methods.

References

1. T. J. Bergin, Computer Aided Software Engineering Issues and Trends for the 1990s and Beyond, Chapter 3, Idea Group Publishing, (1993).

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

3. C. C. Poo and M. K. Chung, Software Engineering Practices in Singapore, accepted for publication in Journal of Systems and Software, (1996).

4. M. Sumner, Factors Influencing the Adoption of CASE in T. J, Bergin, (1993) Computer Aided Software Engineering Issues and Trends for the 1990s and Beyond, Chapter 4, Idea Group Publishing, (1993).

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