Jurnal Teknologi Maklumat & Multimedia 2(2005): 81-103

Standards and Tools in Production and Maintenance of System Documentation

SHAHIDA SULAIMAN, NORBIK BASHAH IDRIS &
SHAMSUL SAHIB UDDIN

ABSTRACT

Implementation of a standard in a software development or maintenance process will provide guidelines on how to conduct the activities in the phases of software life cycle including the documentation to be produced. In addition, the use of Computer-Aided Software Engineering tools or workbenches can automate parts of documenting activities. Despite the importance of standard and tools to be utilised, they are still not widely used. Thus, software engineers still confront with the problems related to documentation particularly system documentation. This paper presents the result of a survey in Malaysia with the main goal to study software engineers' current practice in production and maintenance of documentation based on characteristic, behavior, belief and attitude. Finally, we highlight on what kind of tools should be introduced, why it is introduced and when or how it should be introduced to support the practice.

ABSTRAK

Pelaksanaan sesuatu piawai di dalam proses pembangunan atau pengubah-suaian akan menghasilkan garis panduan bagaimana menjalankan aktiviti-aktiviti di dalam fasa-fasa kitar hidup perisian termasuk dokumentasi yang perlu dihasilkan. Di sam ping itu juga, penggunaanalat kejuruteraan perisian berbantukan komputer atau 'workbenches' boleh mengautomasikan sebahagian daripada aktiviti-aktiviti dokumentasi dengan lebih piawai. Walaupun kepentingan piawai dan alat-alat tersebut boleh dimanfaatkan, tetapi pembangun sistem masih tidak menggunakannya secara meluas. Juruterajurutera perisian masih berhadapan dengan masalah-masalah berkaitan dengan dokumentasi terutamanya dokumentasi sistem. Rencana ini bertujuan untuk membentangkan hasil kaji selidik yang telah dijalankan di Malaysia. Matlamat utama kajian adalah untuk mengkaji amalan semasa juruterajurutera perisian di dalam penghasilan dan pengubahsuaian dokumentasi berdasarkan sifat, tingkah laku, kepercayaan dan sikap. Basil kajian juga dapat mengenat pasti apakah Jmts aoo yang ptrlu .~; keMpllta diperkenalkan dan bila atau bagaimana ia harus diperkenalkan . untuk menyokong amalan terse but.

INTRODUCTION

System documentation (SD) is an important source of software understanding. Having system documentation, a system is easy to maintain. According to Hoffer et al. (1999), SD is detailed information about a system's design specification, its internal workings and functionality. In this study SD includes analysis and design documents. SD should be produced and continuously updated to evolve together with its software. However, most available SD is almost always out-dated or even non-exist at all. On the other hand, a thick set of documentation might be useless too, if such documents do not serve the information required by software engineers particularly during software maintenance. As a result, software maintainers still need to study the source codes to accomplish their tasks.

In order to produce a tool that can serve software maintainers' needs in documentation automation, we believe that the current practice of documentation should be studied. Hence we conducted a survey towards 50 software engineers in Malaysia with the main goal to study their current practice in production and maintenance of SD during software development and maintenance based on four types of data elements: characteristic, behavior, belief and attitude (Kendall and Kendall 1998). We also highlight the questions that argue on what kind of tools should be introduced, why it is introduced and when or how it should be introduced. We would like to emphasise the word "should" instead of "can" because for an instance; what a tool is claimed "can" do is sometimes not actually what a tool "should" do to serve the users' needs. In the following sections we will discuss the background, survey, analysis, findings and finally our conclusion.

BACKGROUND

Programmers spend 40 percent to 60 percent of their time reading the code and attempting to comprehend its logic (Pigoski 1997). Without the assistance of documentation, the percentage may be higher and directly lead to costly software maintenance process. Furthermore, problems in software maintenance are related to how much information is available from the documents especially with respect to the architecture and design of a system. A study stated that documentation being absent, out-of-date or at best insufficient as the third major cause of maintenance problems (van Vliet 2000). While a survey found that the lack of documentation of applications was in the second ranking of the three biggest problems related to software maintenance process (Sousa and Moreira 1998). We highlight the problems related to standard and tools in the following paragraphs.

A lot of CASE (Computer-Aided Software Engineering) tools can support software engineers' activities throughout the whole Software Development

Life Cycle (SOLC). Such fully integrated CASE tools will normally produce a self-generated documentation that will be useful to maintainers. Nonetheless most organisations do not use CASE tools to support all phases of SOLC. "These reasons range from a lack of vision for applying CASE to all aspects of the SOLC to the belief that CASE technology will fail to meet an organisation's unique system development needs" (Hoffer et al. 1999). Hence, software developers might just use a particular CASE tool to draw diagrams that capture user requirements in the analysis and design phase of SOLC but do not use code generators or Reverse Engineering (RE) utility provided. Therefore, there is a lack of integration between the so and the source codes. In addition, RE workbenches can re-document legacy systems by parsing source codes of the subject system and visualise the artifacts in its graph editor. Nevertheless, RE tools are still not widely used for some fundamental difficulties: documents generated are too general or too detail, lack of data needed by maintainers, and RE tools are inflexible (Canfora et al. 1991).

On the other hand, if CASE tools are not used at all, software engineers may need to read through all the source codes again and transform them into graphical notation. "This can be a tedious job, which requires considerable concentration. It is all too easy for concentration to collapse, for even a very short period and miss some vital piece of information" (Lincoln 1993). Besides, during initial development, documentation often comes off badly because of deadlines and other time constraints (van Vliet 2000). Programmers also dislike documenting system as this is seen as a rather boring task compared to the excitements of creation in design and implementation (Macro 1990).

Implementation of a standard such as those from DoD (us Department of Defense) and IEEE (Institute of Electrical and Electronics Engineering) in a software development or maintenance process is a good practice because the standard will provide guidelines on how to conduct the activities in the phases of software life cycle including the documentation to be produced. Some of the standards provide the brief outlines on the contents of specified documentation such as IEEE Std 1016-1998 that recommends practice for Software Design Descriptions but some standards just describe the documentation to be produced in general for instance DoD-Std- 2167 A. However, most organisations in Malaysia do not apply any formal software development methodology (Yahya et al. 2002). Indirectly it shows they probably do not apply any documentation standards to be the guidelines in producing or maintaining so. Therefore documents or so available are of different formats.

Software needs to be maintained and evolves due to new requirements, changes in environment or other factors. Software maintainability is extremely crucial to ensure long lasting software evolution. Documentation is indicated as one of the important factors in software maintainability (Pigoski 1997).

Nevertheless, only the documentation for the first maintenance could be reliable. The link between a program and its associated documentation are sometimes lost during the maintenance process and this may be the consequence of poor configuration management or due to adopting a "quick fix" approach to maintenance (Sommerville 1997).

A number of surveys were conducted to study the problems in software maintenance (Sousa and Moreira 1998; van Vliet 2000) and some surveys such as by Sim et al. (1998) were conducted to study the features required by software engineers in tools for software understanding. However, there is no research has been done to study the role of standards and tools in the current practice. Thus, our study will cover these issues.

THE SURVEY

The survey managed to derive the responses from 50 software engineers from various industries and levels in Malaysia who are involved in software development or maintenance project. Maximum of two software engineers from the same company were allowed to answer the questionnaire. A pilot study of the survey was conducted to 5 software engineers and the questionnaire was refined. The survey was conducted bye-mailing some software engineers who had liaison with the researchers. The questionnaire file was attached to the e-mail and then forwarded to other software engineers. In addition; some identified respondents were personally contacted and distributed with the printed questionnaire by the researchers. This method made the response rate higher and saved cost compared to the method of contacting all organisations in a telephone directory.

The questionnaire was formulated based on the literature review and discussions with some software engineers. It consisted of 35 questions (only 3 of them were open-ended questions) distributed into 3 sections: Section A: Professional Background (AI - A14), Section B: System Documentation and Standard (Bl - B16), and Section C: CASE tools (Cl - C5). Some of closed questions provided "Others" as another option of answers in order to allow respondent to provide the answer that never thought of. Some answers were provided with "No Opinion" option to avoid the feeling of being forced to agree with the answers provided. The questionnaire was quite comprehensive, hence in following analysis and findings we only focus on the issues related to the role of standards and tools in documentation production and maintenance.

THE ANALYSIS

The following discussion will be based on four categories of element or data: characteristic, behavior, belief and attitude. The data was analysed using Statistical Package Software System (SPSS) 9.0 for Windows. The questions without any response were considered missing except for certain filtered questions, which required respondents to skip irrelevant questions.

CHARACTERISTIC

For this element, the survey identified properties of software engineers and so. The respondents were from diverse job positions i.e. from technical to management (see element Crl in Table 1). The data shows 16 respondents were programmers or senior programmers (32%) and 14 respondents (28%) were system analysts. While in management level, the cumulative frequency was 13 respondents (26%). The distribution between the technical and management people surveyed was almost 2 to 1. The respondents were from wide range of industry (element Cr2 in Table 1) with most of them were from software industry (32%), followed by telecommunication (20%), government service (16%) and software consultancy (12%). Other category contributed 20% of those surveyed. In term of experience (refer element Cr3 in Table 1), most respondents had involved in software development or maintenance for 1 to 5 years.

Regarding the usefulness of existing so provided to software maintainers (Table 2, element Cr5), 38 respondents (76%) cited that it was always useful. On the other hand, 7 of them (14%) viewed so provided was not always useful and they *almost* "strongly agreed" with the reasons out-of-date with the mean 4.67, *almost* "agreed" with the reason unreliable (4.17) and incomplete (3.83), "agreed" with the reason misleading (4.00), *almost* "normal" opinion towards low quality (3.20) and "normal" opinion towards unorganised (3.00). When asked whether parts of so produced by CASE tools were always useful (see Table 2, element Cr6), 21 respondents (84%) with CASE tools experience said it was always useful. While only 4 respondents (16%) said that it was not always useful with the "strongly agree" reason (5.00), lack the data needed.

BEHAVIOR

Behavior element investigates on "what organisational members do" during software development or maintenance in relation with documentation. From the response on the number of projects, on average, the respondents are involved in 3 maintenance projects and 2 development projects yearly. Meanwhile, on average, a set of so was produced or modified for both maintenance project and development project per year. Thus, 2 maintenance projects and 1 development project were usually not provided with so yearly.

From Table 3 (element Bv6, question B5, B6 and B7), less than half of the respondents (46%) were provided with a standard or company's own template to write so in which only 5 respondents (22.7%) were provided with documentation standard while the other 16 (72.7%) were only provided with

TABLE 1. Characteristics of software engineers

Element/ Question	Description	Response Item	F	%
Cr1/A1	Job position	Programmer/Senior Programmer	16	32.0
		System Analyst/Senior System Analyst	14	28.0
		Team/Project Leader	7	14.0
		Software/Project Manager	4	8.0
		Software Consultant	2	4.0
		Other	7	14.0
Cr2/A2	Category of industry	Software	16	32.0
		Telecommunication	10	20.0
		Software Consultancy	6	12.0
		Government Service	8	16.0
		Other	10	20.0
Cr3/A3	Experience (regardless of companies)	Less than 1 year	11	22.0
		1 to 5 years	28	56.0
		More than 5 years	11	22.0

Note: F = Frequency, % = Percentage

TABLE 2. Usefulness of SD provided and SD produced by CASE tools

Element/ Question	Description	Response Item	F	%
Cr5/B3	Is SD provided for existing maintenance always useful?	No	7	14.0
		Yes	38	76.0
		Not Relevant	5	10.0
Cr6/C2	Are parts of SD produced by CASE tools always useful?	No	4	16.0
		Yes	21	84.0

Element/ Question	Description	Response Item	Mean*
Cr5/B4	Why SD provided for existing maintenance NOT always useful?	Unreliable	4.17
		Incomplete	3.83
		Out-of-date	4.67
		Misleading	4.00
		Unorganised	3.00
		Low quality	3.20
Cr6/C3	Why parts of SD produced by CASE tools NOT always useful?	Lack the data needed	5.00

Note: F = Frequency, % = Percentage.
*Mean is based on the Likert scale:
1 = Strongly Disagree 2 = Disagree 3 = Normal 4 = Agree 5 = Strongly Agree

company's own template. Most ofthem (17 respondents, 73.9%) claimed that they "always" followed the standard or template provided, 5 respondents (21.7%) cited "sometimes" and 1 respondent (4.3%) cited "never". Regarding the use of CASE tools (element Bv12), 24 respondents (49%) did not use CASE tools, while 20 respondents (40.8%) used CASE tools partially and only 5 respondents (10.2%) used CASE tools for the whole SOLC. Referring to experience in using RE tools (element Bv13), only 3 respondents (6.5%) had ever used the tools compared to 43 respondents (93.5%) without such experience at all.

TABLE 3. The use of SD standard or template, CASE or RE tools

Element/ Question	Description	Response Item	F	%
Bv6/B5	SE provided with standard or template to write SD	No	27	54.0
		Yes	23	46.0
Bv6/B6	Standard or template provided	Documentation standard	5	22.7
		Company's own template	16	72.7
		Other	1	4.5
Bv6/B7	How frequent the standard or template provided is followed	Always	17	73.9
		Sometimes	5	21.7
		Never	1	4.3
Bv12/C1	Use CASE tools for the whole SDLC	None	24	49.0
		Yes	5	10.2
		No, only parts of SDLC	20	40.8
Bv13/C4	Use single reverse engineering tool	No	43	93.5
		Yes	3	6.5

Note: F = Frequency, % = Percentage

Regarding the software packages used to produce or maintain so, the responses are ranked by the mean values and the usages are compared, as in Table 4. Based on the table, it reveals that the respondents who used CASE tools for tb& whole SOLC used most of the software packages; both word processor and spreadsheet (3.00) and graphical tools (2.33), compared to the other two groups. For the other two groups ("None" and "No, only part of SOLC"), both mostly used word processors (2.81 and 2.79 respectively). The former group used spreadsheet application more (2.08) than the latter group (1.50), but used less graphical tool (1.62) compared with the latter group (2.09).

Table 5 reveals that respondents without documentation standard or template, produced or maintained less number of so (0.88 for development and 0.80 for maintenance projects) and they faced more maintenance projects without so yearly (1.96). Meanwhile, the respondents who were provided

TABLE 4. The use of CASE tools versus the use of software package

CASE Tools Usage For Whole SDLC		Word Processor	Spread Sheet Application	Graphical Tool	Other Tool
None	Mean	2.81	2.08	1.62	2.00
	N	21	12	13	1
	Std. Deviation	.51	.67	.77	-
Yes	Mean	3.00	3.00	2.33	3.00
	N	4	1	3	1
	Std. Deviation	.00	-	.58	-
No, only part of SDLC	Mean	2.79	1.50	2.09	2.67
	N	19	10	11	3
	Std. Deviation	.42	.71	.70	.58
Total	Mean	2.82	1.87	1.89	2.60
	N	44	23	27	5
	Std. Deviation	45	.76	.75	.55

•

TABLE 5. The use of standard or template versus the number of SD and maintenance project without SD

Provided with Standard or Template to Write SD		Number of SD Produced for New S/W Development Project	Number of SD Produced /Modified for S/W Maintenance Project	Number of S/W Maintenance Project Not Provided with SD
None	Mean	.88	.80	1.96
	N	25	25	24
	Std. Deviation	.97	1.22	5.51
Documentation Standard	Mean	1.40	1.00	1.60
	N	5	5	5
	Std. Deviation	1.14	.71	1.52
Company's own template	Mean	2.38	2.38	1.08
	N	13	13	12
	Std. Deviation	3.73	2.87	1.51
Other	Mean	.00	1.00	1.00
	N	1	1	1
	Std. Deviation	-	-	-
Total	Mean	1.36	1.30	1.64
	N	44	44	42
	Std. Deviation	2.24	1.92	4.25

with documentation standard produced and modified more SD (1.40 for development and 1.00 for maintenance projects) and faced less maintenance projects without SD (1.60) as compared to the former group. In addition, those with companies' own template, produced and maintained the most number of SD yearly (2.38 for both development and maintenance projects) and the least number of maintenance projects not provided with SD (1.08). We do not discuss on the "other" group.

The number of SD produced for new software development (see Table 6) is the highest (1.71) among software engineers who did not use CASE tools at all followed by those partially used CASE tools (1.16) and fully used CASE tools (1.00). On the other hand, the highest number of SD produced or modified in software maintenance project contributed by software engineers who partially used CASE tools (1.63) followed by those with no CASE tools (1.29) and with CASE tools (0.50). Regarding the number of maintenance project was not provided with SD, this phenomenon mostly occurred among software engineers partially used CASE tools (2.61), while those fully used CASE tools had the mean value 1.50 compared to only 0.95 of those without CASE tools.

TABLE 6. The use of CASE tools versus number of SD and maintenance project without SD

Use of CASE Tools For Whole SDLC		Number of SD produced for new s/w development project	Number of SD produced/modified for s/w maintenance project	Number of s/w maintenance project not provided with SD
None	Mean	1.71	1.29	.95
	N	21	21	20
	Std. Deviation	.2.97	2.00	1.28
Yes	Mean	1.00	.50	1.50
	N	4	4	4
	Std. Deviation	1.41	.58	2.38
No, only part of SDLC	Mean	1.16	1.63	2.61
	N	19	19	18
	Std. Deviation	1.34	2.03	6.25
Total	Mean	1.41	1.36	1.71
	N	44	44	42
	Std. Deviation	2.25	1.93	4.24

ATTITUDE

The element of attitude covers the issue of "what people in the organisation say they want in a document generator or RE tools". This section will investigate whether it is true that software engineers prefer graphical to textual software representation in order to understand the software, and what features are in favour that should be incorporated into a document generator tool.

The analysis result is shown in Figure 1. The figure illustrates the difference in the need of textual description versus graphical representation, in which the mean of graphical representation of system is higher compared to that of textual description. The mean of textual description of system architecture or system and subsystem is 3.18 while 3.83 for the graphical representation. The mean of textual description of modules or programs and their relationships is 3.20 but 3.83 for the graphical representation. Meanwhile, the mean of textual description of data flow is also lower (3.15) if compared to that of data flow graph (3.72). The mean for textual description of procedures or methods or functions is 3.27 whilst the mean for program versus file cross-reference table is 3.14. The most needed features among all are graphical system-subsystem flow and graphical representation of components' relationships, which share the same mean value (3.83).

For other features like search utility, documentation layout generation, applicatioiiIiiwide range of language, interactive browsers, and link between source code and graphical representation; their means are 3.20, 2.93, 2.98, 2.93 and 3.29 respectively. All the features have the means within the range of 2.00 (Less Required) to 4.00 (Required) which indicate that all features are basically necessary to have but not the most demanded in the respondents' point of view.

T-test was used to study the significance of difference in means. The null hypothesis is "there is no significant difference between the means of textual and graphical software visualisation". Table 7 reveals that all the three pairs are able to reject the null hypothesis (less than 0.025).

BELIEF

The belief element ponders the issues related to what software engineers think about the importance of SD, its standard or template, reasons to have documentation standard or template, and reasons for not producing SD. Element Bfl (see Table 8) reveals that the majority of respondents (33 respondents, 68.8%) strongly agreed that SD was important, while 14 respondents (29.2%) agreed, and only one respondent (2.1 %) was normal on the issue. Regarding the importance of documentation standard or template, element Bf3, majority of them stated "Yes" (98%) and strongly agreed with the reasons to provide guidelines (4.64), communicate critical information (4.56), communicate necessary information (4.53), standardise format of documents (4.53) and organise documents (4.46). Surprisingly, there was a respondent who was against the belief and stated "No".

Features / Need	1	2	3	4	5
Textual description of system architecture/system and subsystem			3.18•		
Graphical representation of system and its subsystem				•3.83	
Textual description of modules/programs & relationships			3.20•		
Graphical representation of components' relationship i.e. call graph				•3.83	
Textual description of data flow between modules			3.15•		
Graphical representation of data flow between modules i.e. data flow graph				•3.72	
Textual description of all components			3.27•		
Program versus file cross reference table/file usage table			3.14•		

Other Features	1	2	3	4	5
Search utility to identify affected components in maintenance task			•	2.93	
Generate documentation layout conform to company's standard			•	3.20	
Can be applied in wide range of language and platforms			•	2.98	
Read textual / graphical representation using interactive browsers			•	2.93	
Link between source code and its graphical representation			•	3.29	

Note: 1 = Not Required, 2 = Less Required, 3 = Normal, 4 = Required, 5 = Most Required
Graphical feature: ——— Textual feature: ———

FIGURE 1. The need of graphical versus textual features and other features

TABLE 7. T-test of textual versus graphical features

		Paired Difference							
		Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference		t	dt	Sig. (2-tailed)
					Lower	Upper			
Pair 1	Textual Description of System Architecture / System-subsystem – Graphical system-subsystem flow	-.62	1.54	.23	-1.09	-.16	-2.71	44	.010
Pair 2	Textual Description of Modules/ Programs and their relationship – Call graph	-.63	1.29	.19	-1.01	-.25	-3.32	45	.002
Pair 3	Textual Description of System Data/ Data Flow – Data flow graph	-.57	1.05	.15	-.88	-.25	-3.66	45	.001

TABLE 8. Importance of SD, standard and template

Element/ Question	Description	Response Item	F	%
Bf1/B1	Importance of SD	Normal	1	2.1
		Agree	14	29.2
		Strongly Agree	33	68.8
Bf3/B8	Importance of documentation standard or template	No	1	2.0
		Yes	48	98.0

Element/ Question	Description	Response Item	Mean
Bf4/B9	Reasons there should be documentation standard or template (If "Yes" in Question B8) information	Communicate necessary information	4.53
		Communicate critical	4.56
		Provide guidelines	4.64
		Organise documents	4.46
		Standardise format of documents	4.53

Note: F = Frequency, % = Percentage.
Element Bf4 based on the Likert scale:
1 = Strongly Disagree 2 = Disagree 3 = Normal 4 = Agree 5 = Strongly Agree

Figure 2 illustrates the reasons for not producing SD. The top three reasons were time constraints (4.30), commercial pressures (3.80) and SD not requested by project leader or software manager (3.60). The other reasons were because SD was not requested by customer and it was a tedious task, which share the same mean value (3.40), costly to keep updated (3.30), a boring task (3.20), done by other people and not interested, both with the same mean (3.00). These mean values were based on Likert scale: I = Strongly Disagree, 2 = Disagree, 3 = Normal, 4 = Agree and 5 = Strongly Agree.

We compare the means based on the Likert scale, of the top five reasons (see Figure 2) for not producing SD with the respondents' belief towards the importance of SD as in Table 9. We do not discuss on the "Normal" group, but only on the "Agree" and "Strongly Agree" groups. The mean for "time constraints" reason is the highest for both groups compared to other reasons in each group. But, comparing between the two groups themselves, the mean of "Strongly Agree" group is 0.43 higher than that of "Agree" group. The second highest reasons for "Agree" group are "not requested by project leader or software manager" and "not requested by customer" (sharing the mean 4.00 of scale "Agree"). On the other hand, the "Strongly Agree" group cited the second top reason as commercial pressures (3.86), followed by three other reasons: "tedious task" (3.47), "not requested by project leader or software manager" (3.13) and "not requested by customer" (2.87). While for those "agreed" on the importance of SD, the last two reasons were "commercial pressures" (3.75) and "tedious task" (3.42).

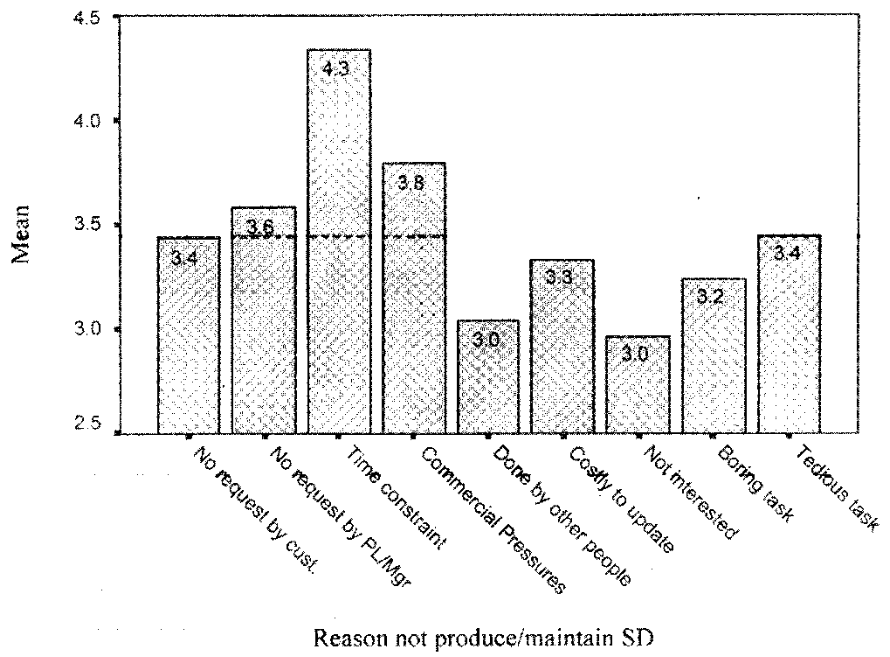

FIGURE 2. Bar chart of the mean values versus reasons for not producing or maintaining SD

THE FINDINGS

Software engineers confront with more maintenance projects without SD compared to development projects with the ratio of 2: 1. Despite their perspectives on the importance of SD and standards to be followed (see Table 8), both software development and maintenance projects suffer from non-existence of SD. On the other hand, software engineers tend to appreciate existence of MY kinds of SD. This is because majority of them cited that SD provided for existing software maintenance or parts of SD produced by CASE tools were always considered useful to them regardless of their quality (see Table 2).

Reasons for not producing or maintaining SD range from management to technical issues. As stated by van Vliet (2000), the foremost reason for not producing or maintaining SD is due to time constraints. This was inline with our findings (Figure 2). We discuss only on the top five reasons and the mean comparison according to the belief towards the importance of documentation.

TABLE 9. Importance of SD versus reasons for not producing SD

Importance of SD		Time Constraints	Commercial Pressures	Not Requested By Project Leader or S/W Manager	Not Requested By Customer	Tedious Task
Normal	Mean	5.00	3.00	3.00	4.00	3.00
	N	1	1	1	1	1
	Std. Deviation	-	-	-	-	-
Agree	Mean	4.07	3.75	4.00	4.00	3.42
	N	14	12	13	13	12
	Std. Deviation	1.14	1.36	1.08	.82	1.31
Strongly Agree	Mean	4.50	3.86	3.13	2.87	3.47
	N	20	14	15	15	15
	Std. Deviation	.61	.86	1.41	1.19	1.30
Total	Mean	4.34	3.78	3.52	3.41	3.43
	N	35	27	29	29	28
	Std. Deviation	.87	1.09	1.30	1.15	1.26

Software is always developed or maintained within a specified schedule and must be installed at customers' site as planned. This provides time constraints and causes commercial pressures to software engineers. They tend to emphasise more on producing the software and forget about the documentation. The third and fourth ranked reasons were: not requested by their superiors and customers respectively. Most customers are not aware of the need to have documents as a complementary product to a software system. Since customers do not request documentation, project leaders or software managers do the same to their software engineers although they believe in the importance of so to be produced or maintained together with the software system. Sharing the fourth ranked reason previously mentioned was the reason why software engineers did not produce or maintain so as it was a tedious task.

Technically, documenting activities are tedious particularly when we need to ensure the link between source codes and documents is always updated. In addition, software engineers with stronger belief towards the importance of so, tend to have more concrete reasons ("commercial pressures" and "tedious tasks") besides the foremost reason "time constraints" (Table 9). For this group of software engineers, they were *almost* "normal" towards the reasons "not requested by project leader or software manager" and "not requested by customer". Thus, it reveals that these software engineers have better perspectives towards documentation, and do not simply neglect production of documentation or give a weak reason such as "Although I know I should do, I don't do because I'm not requested to do".

Single or integrated CASE tools that cover the whole SOLC (like Oracle Designer and Developer, Rational Rose integrated with Visual Basic) should be able to lighten software engineers' work hence eliminate the problems related to so. Most development tool suite or package's latest version like Visual Basic and Borland JBuilder provide the utility to document components of a software system in the development environment itself. As a result, the use of CASE tools (fully or partially) can make software engineers assume the data they feed into a CASE tool during software development can be referred as an alternative to documentation especially if they employ rapid application development or extreme programming in their software development or maintenance. This is supported by the fact that most of them (21 of 25 software engineers with full or partial CASE tools experience) found the parts of documents provided by CASE tools were useful (Table 2).

On the. other hand, software engineers without CASE tools tended to put best effort to produce documents in development projects (Table 6). The same scenario occurred when producing so for software maintenance in which software engineers who used CASE tools for the whole SOLC produced or maintained less so as compared to the other two categories. Consequently, software engineers who used CASE tools (fully or partially) faced a higher number of software development or maintenance projects without so compared

with those without CASE tools. In this case, the use of CASE tools does not seem to be able to solve problems in SO as always expected. Besides the use of CASE tools, software engineers used software packages like word processor, spreadsheet application and graphical tool to produce or maintain documents (Table 4). We expected those without CASE tools or partially used CASE tools, used software packages the most. But our study shows that the two groups used less software packages if compared to the group with CASE tools for the whole SOLC. Based on previous discussion on the use of CASE tools, software engineers without CASE tools or partially use it tend to produce less so, hence they also use less software packages to document software system.

Software engineers believed in the importance of so particularly to provide guidelines and also the enforcement of standard or template while writing so (Table 8). Despite the belief, more than half of software engineers were not provided with standard or template by their companies (Table 3). Only 5 software engineers were provided with documentation standard while 16 software engineers were provided with company's own documentation template. When enforced to follow the standard or template in writing SD, most software engineers tended to "always" follow them and only some software engineers "sometimes" followed them (Table 3). In addition, software engineers with documentation standard or template tended to produce or maintain more so compared to those without it (Table 5). Consequently, the former group of software engineers faced less maintenance project without so compared to the latter.

Some CASE tools like Rational Rose provide the utility to reverse engineer existing source codes into the design level. However, this tool will work well if software engineers start with the analysis and design using the tool and the coding is done using the integrated development tool. Otherwise, the tool will visualise the class diagrams only without detail information on the relationships of the components such as the function calls. Rational Rose also provides a documentation environment called SOOA but this utility is not linked with RE utility. Hence documentation still must be done manually (with some automation of components' description) to capture the diagrams done during analysis and design. Despite the existence of single RE tools or workbenches, they are still not widely used in Malaysia. In fact the difference between RE tools and CASE tools with RE facility is still unclear. For instance there were two respondents specified Visual Modeler (wrongly thought as single RE tool, in fact it is just like Rational Rose but specially dedicated for Visual Basic) and Rational Rose respectively, as a single RE tool experienced. Only three out of 46 respondents claimed to ever use a single RE tool. One of them specified CASE Tool 2 (never identified in our literature review) while another two software engineers did not specify the tools' names. Hence the three respondents' experience towards RE tool is still a question.

RE or document generator tools should have the features required by software engineers otherwise the data served by the tools might not be useful. The study discovered that graphical representation was preferred more than textual representation, and the difference in the three types of features identified in the study (system-subsystem architecture, data flow graph and call graph - see Figure 1) was significant (Table 7) for each feature compared to its textual counterpart.

WHAT, WHY, WHEN AND HOW TOOLS "SHOULD" SUPPORT THE PRACTICE

From the findings we summarise what, why, when and how tools "should" support (not "can" support) the practice in production and maintenance of SD.

a) *What tools should support?*

Both documentation and enforcing a standard while wntmg it are important. Thus, a tool that can, visualise the software artifacts particularly with features specified by software engineers in this study, and some features highlighted in our previous work (Sulaiman et al. 2002) into a standardised documentation template should be introduced. The toot should focus on production of documents related to the analysis and design of a system, which are crucial to software maintainers.

b) *Why tools should support?*

From the reasons for not producing or maintaining SD, in our study, it shows that tools should support documentation because software engineers frequently face time constraints and commercial pressures in their projects. The tools should also attempt to make documenting activities interesting, easy and fun in order to eliminate problems related to negative perspectives towards documenting activity.

.

c) *When tools should support?*

From the findings in this study, at least one set of SD was not produced during development stage yearly. Hence, it shows that tools to automate documenting activities should be introduced since software development, and should be continuously utilised in the following maintenance . processes. Regardless of what tools software engineers use to perform analysis, design and coding, the tools should be an alternative to produce the most updated SD. In an SDLC of a software development or maintenance process, the tools should be used after each implementation phase when there is no more changes to be made, in other word, when the version or revision of the software has been released.

d) *How tools should support?*

If document generator tool is introduced early in the development stage, the tool will be able to capture the knowledge from software developers regarding the clustering of the components in the software. Subsequently, the written source codes can be parsed during reverse engineering process, and the captured software artifacts can be visualised or viewed and also printed into a standardised document template, configured and archived into softcopy for future use. If the tool is used only when software is on a maintenance stage, software engineers need to study the existing software components to suggest how to cluster them before other steps can be taken. This may eliminate the weaknesses in the current tools studied by Sulaiman and Idris (2002).

Therefore there is a difference between "can" and "should" as discussed earlier. A tool that is claimed "can" perform certain task might not satisfy users' needs if they do not accomplish the tasks they "should" perform. Based on the above arguments, we illustrate how CASE tools or workbenches, and also documentation standards can play their roles within a software life cycle as illustrated in Figure 3.

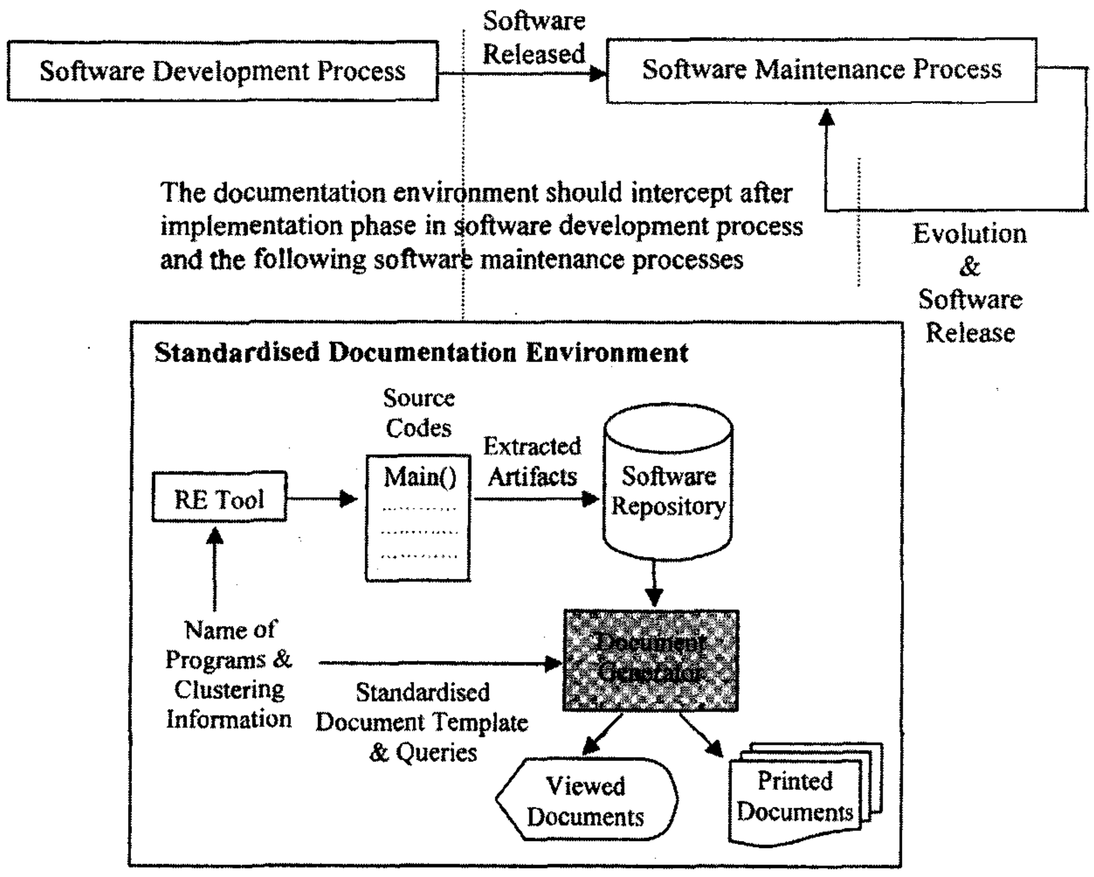
Software Development Process
Software Released
Software Maintenance Process
The documentation environment should intercept after implementation phase in software development process and the following software maintenance processes
Evolution & Software Release
Standardised Documentation Environment
Source Codes
Main()
Extracted Artifacts
Software Repository
RE Tool
Document Generator
Name of Programs & Clustering Information
Standardised Document Template & Queries
Viewed Documents
Printed Documents

FIGURE 3. The implementation of a standardised documentation environment within a software life cycle

The dotted lines show the phases when the documentation environment should intercept. At these levels, source codes are in the most reliable and updated condition where there is no more changes and software is released. In the documentation environment, the shaded rectangle represents a document generator which is developed using an enhanced approach of software visualisation (as discussed in a previous work by Sulaiman and Idris 2002). Prior to generating a document, a user is required to feed in the name of programs and clustering information of the programs specified. The existing parser will parse the source codes and then the software artifacts will be retained in a software repository. The document generator will extract related information from the repository including the clustering details that are tagged with the parsed software components and generate the documents to be viewed and printed based on the document template or queries specified by the user.

CONCLUSION

The study implies that both standards and tools play very significant roles in order to improve the current practice in production and maintenance of documentation particularly system documentation. Despite software engineers' belief on the importance of SD, they still do not produce SD during software development or maintain SD during software maintenance for the top two reasons: time constraints and commercial pressures. Besides, the use of software packages, CASE tools or workbenches like RE, does not provide total solution to the problems related to SD. However the use of both standards and tools in an organisation tends to increase the number of SD produced or maintained. Although an organisation does not employ any documentation standard, software managers should at least provide a documentation template and then they can slowly enforce a suitable standard in their documentationrelated activities.

On the contrary, a tool may be useless if it cannot satisfy users' needs. Thus we emphasise on what, why, when and how tools should support the practice based on our study, to provide a guidance to CASE tools developers. In a nutshell, we believe the documentation practice can be improved by using a document generator tool integrated with a RE technology in a standardised documentation environment to assist software engineers in documenting critical information during development and the following maintenance processes, that is after an implementation phase in which source codes are at the most updated level. Nevertheless, the functionalities of such tools are still not widely understood and utilised by software engineers in Malaysia. Hence we should increase the awareness of the need for the effective tools to improve their current practice in production and maintenance
of documentation in general, and SD in particular.

REFERENCES

Canfora, G., Cimitile, A. and Carlini, U. 1991. A logic-based approach to reverse engineering tools production. *IEEE Transactions on Software Engineering*

Hoffer, J. A., George, J. F. and Valacich, J. S. 1999. *Modem systems analysis and design.* 2nd Ed. Reading, USA: Addison Wesley.

Kendall, K. E. and Kendall, J. E. 1998. *System analysis and design.* 4th Ed. London: Prentice Hall.

Lincoln, A. D. 1993. Computer-aided documentation for software maintenance. *lEE Colloquium on Issues in Computer Support for Documentation and Manuals* 169(7): 1-3.

Macro, A. 1990. *Software engineering concepts and management.* Hertfordshire, UK:
Prentice Hall.

Pigoski, T. M. 1997. *Practical software maintenance: best practices for managing your software investment.* Indianapolis, USA: John Wiley.

Sim, S. E., Clarke, C. L. A. and Holt, R. C. 1998. Archetypal source code searches: a survey of software developers and maintainers. *Proceedings of the Sixth International Workshop on Program Comprehension,* 24-26 June. Ischia, Italy, 180-187.

Sommerville, I. 1997. *Software engineering.* England: Addison Wesley.

Sousa, M. 1. C. and Moreira, H. M. 1998. A survey on the software maintenance process. *Proceedings of the International Conference on Software Maintenance,* 16-20 November. Washington, USA, 265-274.

Sulaiman, S. and Idris, N. B. 2002. An enhanced approach of software visualization in reverse engineering environment. *Proceedings of the National Conference on Computer Graphic and Multimedia (CoGRAMM'02),* 7-9 October. Melaka, Malaysia, 459-464.

Sulaiman, S., Idris, N. B. and Sahibuddin, S. 2002. A comparative study of reverse engineering tools for software maintenance. *Proceedings of the 2nd World Engineering Congress,* 23-25 July. Sarawak, Malaysia, 478-483.

van Vliet, H. 2000. *Software engineering principles and practice.* 2nd Ed. New York, USA: John Wiley.

Yahya, Y., Mohd. Yusof, M., Yusof, M. and Omar, N. 2002. The use of information system development methodology in Malaysia. *International Journal of Information Technology* 2: 15-34.

Shahida Sulaiman
Pusat Pengajian Sains Komputer
Universiti Sains Malaysia
11800 USM Pulau Pinang Malaysia
e-mail: shahida@cs.usm.my

Norbik Bashah Idris, Shamsul Sahibuddin
Centre for Advanced Software Engineering (CASE)
Universiti Teknologi Malaysia City Campus
Jalan Semarak
54100 Kuala Lumpur
Malaysia e-mail:norbik@case.utm.my.shamsul@case.utm.my