Suggested Functionalities of a Software Maintenance Tool for a University Application System

Hidayah Sulaiman
Informatics Department
Universiti Tenaga Nasional
Malaysia

Rusli Abdullah
Faculty of Computer Science and Information Technology
Universiti Putra Malaysia
Malaysia

Abstract
Using the right type of tool for maintenance activity and application support is essential in order to meet today's technological demands. These days, almost all business operations rely on application system efficiency. Maintenance is also seen as the highest percentage of work activity done in a software development life cycle as many software practitioners and researchers have quoted. Even in 1991 there was already a study by Abran et.al that showed a mere increased in the maintenance activity and support to users. This paper will present the result of a study done on various public and private universities of Klang Valley in Malaysia. Based on the findings, the maintenance tool is seen as helpful and useful for the developers or maintainers of a university application system to assist them in their daily maintenance activity.

Keywords:
Software Maintenance Tool, First level support, second level support, knowledge base

1. INTRODUCTION

Software maintainers have long been acutely aware of the challenges involved in managing software change processes (Dean Jin, 2005). Activities such as source code manipulation, change, testing, documentation, change management and user support rely heavily on the analysis and understanding of the complex system structure of both legacy system and modern software system. Jin (2005) claims that it is widely accepted that tools that support software analysis and maintenance tasks would go a long way towards addressing the constraints that the software developer and maintainers work with on day-to-day basis. As defined by Lethbridge & Singer (1997) tools can be everything from large scale integrated CASE products to simple one-function command or features. It includes anything functional that can help software maintainer to solve their maintenance problem.

However, despite having a lot of available tools in the market that can assist the maintenance activity, not many are being used by practitioners namely in a non-software development organizations. Very few are considered essential for a particular development or maintenance task. The chosen tools must support program understanding and reverse engineering, testing, configuration management, and documentation (Takang and Grubb [1996]).

Manuscript received March 5, 2008
Manuscript revised March 20, 2008

2. AIMS

The purpose of this study is to identify the functionalities of a software maintenance tool that can be of assistance to maintainers of a University application system. In order to achieve this, the following objectives are to be satisfied:

- To establish what software maintenance tool is and why a software maintenance tool can be increasingly helpful for daily maintenance activity.
- To identify and assess the functionalities of a software maintenance tool.
- To suggest appropriate functionalities of software maintenance tool that could be used within the university on a specific application system.

University environment is different as compared to a real software development organization. The structure of the resources handling the system or application may not be as well-defined and designated to a specific group of people.

Although Information Communication Technology (ICT) department is well established in the Universities, person involved in maintenance of the application could also be the developer of the application. In conjunction to that, some of the applications are also outsourced to software vendors; therefore one designated person is enough to be assigned to monitor the application.

3. METHOD

The focus is given to 9 public and private Universities located in the Klang Valley of Malaysia. Each of these Universities has a common core business application that each is maintained by one or two developers/maintainers. This project focuses on three main application actively maintained by the ICT department. The core business application focused may differ from one University to another depending on which application is being highly maintained. The common applications are Student Management System, Human Resource Management System, Billing and Finance.

Since the 3 core applications are being focused on each university, the sample study of 40 application maintainers is targeted. A preliminary round of information collection was conducted to 2 universities in order to observe the daily routine of an application maintainer. Questions were asked regarding the application and job activity of a software maintainer.

Further study was done through literature of related work via journals and conference paper as well as an observation to a support management system of a multi-national organization. All the information are gathered and compiled to be transformed into a set of questionnaires to be distributed to the software maintainers of the universities. Respondents filled out the questionnaire at their designated office. Each respondent receive a booklet consisting of:

1) A cover sheet explaining the purpose of the study, brief explanation of each section and the name of the person conducting the study.
2) The demographic questions
3) Functionality of existing software maintenance tool. The purpose of this section is to identify the level of relevance of the current available functionality of software maintenance tool proposed and published by researchers and practitioners.
4) Proposed Functionality of a Suggested Software Maintenance Tool. The purpose of this section is to propose a model of a software maintenance tool through its functionality. The functionalities are derived from the study carried out on various previous works done as well as interviews and observation to a higher education institution and corporate organization.

Respondents were also guaranteed anonymity and confidentiality of their responses and they were told that the session will take about 30 – 40 minutes.

4. RESULTS AND DISCUSSION

4.1 Description Of The Sample

The chart in Figure 4.1 below denotes the result from the sample on the current position that best describes the respondents. The developers make the highest job category, 54.05%, followed by the Project/Team Leader, 40.54% and manager, 13.51%. Both support engineer and other positions such as System Analyst contributed 8 % and 4% respectively.

Position	Percentage
Project/Team Leader	40.54
Manager	13.51
Developer	54.05
Support Engineer	8
Others	4

Figure 4.1: Current Position

These figures are relevant as all of the universities do not have a specific person assigned to a specific job position; developers, support engineers and sometimes the team leaders are actually the same set of people involved in the maintenance activity.

The maintainers are currently involved in software activities such as software requirements, software quality assurance, software design, configuration management, code and unit testing, maintenance, test and integration and others such as software upgrade.

The next question was to ask on the tasks that they carry out daily and the highest category of job activity falls under maintenance, 81.08%, followed by test and integration, 70.27%, software design, 64.86%, software requirements and code and unit testing, 59.46%, change management, 43.24% and others such as upgrades, 10.81%. The results yield a very strong influence towards answering the rest of the sections in the questionnaire as people in the maintenance area are highly needed in providing answers to questions related to the maintenance tool functionality.

For the current and overall software experience category, the highest range of current software experience comes from people with 3 to 8 years of experience. This is acceptable as people with 3 to 8 years of experience are considered those who are already comfortable at maintaining the current application and have the in depth knowledge on the application that they are maintaining. Therefore, it would be easy for them to understand and provide relevant answers.

Respondents were asked if they had any experiences of doing maintenance with assistance of a tool, and only 67.6% of them responded that they had that sort of experience. Then again, this complies with the definition by Lethbridge & Singer (1998), tools is defined as everything from large scale integrated CASE products to simple one-function command or features. Some of the tools mentioned are SILA for database management, my-Genie for request ticket problem, remote desktop, LEKO, testing script and simple modules for amendments, testing, and performance monitoring. There is no specific integrated tool mentioned by the respondents that is designated to assist them in doing support or maintenance activity to the application system.

4.2 FUNCTIONALITY OF EXISTING SOFTWARE MAINTENANCE TOOL.

The functionality is gathered from different industrial environment and it is hoped that the answers provided will show relevance of these functionalities towards maintaining a University application in the higher education environment. All the questions compiled for this section are obtained from the literature review done on the topic of software maintenance tool. There are nine questions in this category and some has its sub-category pertaining to maintenance tool and activity.

The first question asked was whether they are using object oriented approach to maintain their application and will these features be relevant to their job:

a. Abstract control flow of the application.
b. Produce Object maps
c. Produce Inheritence Map

The responds to this question shows that not all of the universities are using the object oriented approach in doing maintenance to their application. Question 2 asks the maintainers on their opinion, should they have a tool that caters for change management and support request; will these features assist them in the software maintenance tool:

a. Issue date and arrival time of the request
b. Owner of the request and their department
c. Description of the problem
d. Request priority
e. Sample answer codes to solve the request from the tool
f. Status of request
g. Response time in handling the request
h. Closed date for request

All of the respondents agreed that the issue date and arrival time of the request is very important to a tool that caters for change management and support request. This is also followed by the feature of having the description of the problem. These two features are seen as crucial and important to be implemented in a software maintenance tool. Status of request is seen as the following top feature, followed by having the owner of the request and their department as well as request priority. Some maintainers feel that information on the department or the person is not of high priority for them. As long as the description of problem is there and it is valid, that should be sufficient.

However, as 78.4% agreed that this information is essential as they would need to get back to the owner in order to have a clearer picture on the problem. This is true as based on their experience some non-technical user might not provide the correct description of the problem or might not even know where the problem lies. As for the request priority, not all voted "yes" as some maintainers feel that request priority should be decided by the maintainer themselves as they know how much time and effort should be done on the problem. They feel that some impatient users might put a high priority on all problems even though it is of a minor one. Then again 78.4% of them have voted "yes" as they feel that it is important to categorize the problems on its criticality in order for them to plan and prioritize their job. The next two features which are considerably important is the closed date for request and response time in handling request.

The functionality on having sample answer codes to solve the request from the tool is seen as having the lowest priority in which 45.9% answered "yes". They require this feature to assist them in expediting their maintenance activity. As this may not be of a practice to the other 43.2% of as they feel that it is not needed due to knowing inside out of the application being maintained. Question 3 of this section asks the respondents' on should a software maintenance tool be sub-categorized into:

a. User-support module: contains helpdesk module and able to generate report for the support activities.
b. Repair: a module to do correction of the application
c. Enhancement: make changes to the application according to new data requirement and enhancement of the application performance

Two out of the three sub categories obtained more than 80% of "yes" votes. Therefore it is agreed that a standard software maintenance tool should consist of repair and enhancement. As for the user-support module, 75.7% has answered yes to this feature which is considerably acceptable and strong for this feature to be included in a software maintenance tool.

For the response on should a software maintenance tool be equipped with a knowledge base system that consists of a bug tracking database to find similar existing problems and able to communicate across different groups, 83.8% of the respondents voted "yes". This feature is seen as very helpful in assisting their maintenance activity especially in expediting the problem solving area. Next is to obtain response on the general features that has been requested by some of the software practitioners in other industries. The features are:

a. Module to do modeling and manipulating of source code
b. Designing module
c. Debugging tool
d. Finding memory leaks
e. Source code exploration tool
f. Analyze coding and naming conventions
g. Web front End/ Interface
h. Automated testing tool
i. Tool to ease installation or upgrade.

Out of the 9 features, the debugging tool and Web front end/interface yields "yes" votes of more than 90%. This can be seen as an essential feature that must be included in a tool. Besides that, features like designing module, automated testing tool and tool to ease installation and upgrade holds more than 75% of "yes" votes. This is seen as needed but not as crucial as the one before. The rest of the features such as source code exploration tool, finding memory leaks and module to do modeling and manipulating of source code is voted yes by 70.3%, 51.4% and 45.9% respectively.

With respect to the usage of a CASE tool, respondents were asked if the following features are required in their daily job routine:

a. Documentation
b. Designing of system
c. Create requirement analysis
d. Draw diagram
e. Planning of system
f. Code generation
g. Prototype development
h. Project management and tracking
i. Change management

It seems not many universities are actually using CASE tools. However, the highest usage of CASE tool fall into the designing of system and drawing diagram, code generation, and change management.

On the subject of documentation, respondents were asked on whether they have a well-written documentation that can adequately support the changing needs and strategies when modifying their application. For this, the results showed 56.8% of them responded "No". Subsequent to that they also agreed that it will be a great assistance to have a tool that incorporates documentation on the application system being maintained. In conjunction to that they were asked a final question regarding documentation that should these be part of the module incorporated in the tool, would they find it useful:

a. A window that contains graphical representation of the application call structure (functions of the application depicted in a hierarchical manner).
b. Window that contains the source code for different subroutines
c. Window that contains variable information such as name, description, size.
d. When a variable is clicked, the item is highlighted in all windows that have relation to the variable and you can see the occurrences of the chosen identifier.
e. There is an edit window for you to edit your source code.

From the overwhelming respond, these are seen as most important in a documentation module.

4.3 Functionality of a Suggested Software Maintenance Tool.

Through the literature done for section 2 of the questionnaire, section 3 was created in order to propose a model of a software maintenance tool through its functionality. It is hoped that the functionality suggested in this section for the software maintenance tool is able to assist the maintainers' daily maintenance activities and bring about usefulness in their work. The functionalities are sub divided into two major maintenance activities:

- Support, Change Management and Knowledge Base
- Documentation

From the analysis, these are the functions of the suggested maintenance tool:

1. Request creation for existing user
2. Request creation for new user
3. Problem assignment
4. Request-solution timer
5. Knowledge Base update
6. Create formal answer to user
7. Request status determination
8. Request escalation to SLS
9. View problem/request statistics
10. View graphical functional flow of application maintained
11. View source code
12. Search variable

4.3.1 First Level Support (FLS)

FLS will be the main communicator between the user of the system maintained and the ICT department. FLS will be the first point of contact for the support call as they will take down the problems via phone, email, fax or SMS. They are also responsible to understand the problems raised by the users as they need to determine the severity of the problem and who should solve the problem. The FLS will also need technical knowledge to solve some of the problems raised and able to access the knowledge base to find similar problem solution to solve the problems raised by the users. The FLS does the remedy and update the knowledge base with the solution and is responsible in providing solution to the user and making sure that the answer is accepted or rejected.

However, there are pros and cons of this scenario as the first level support might have to key in problems as their daily routine instead of solving the problems but nevertheless the problems entered by users might not portray the clear picture of the problem and cause confusion to the maintainers.

4.3.2 Second Level Support (SLS)

The SLS is involved in the maintenance activity should the FLS is not capable of solving the problem or the severity of the problem raised by the user is at a very high status. The SLS does the remedy and restoration as well as updating the knowledge base and close the case by providing the answer to the team leader.

4.3.3 Team Leader/Manager

The team leader is in charge of assigning the problems escalated by the FLS to the SLS. They review the solution and provide an official answer to the FLS who escalated the problem. The team leader is able to view the time taken by each of his team members to solve each support call. They are also able to view the number of solved cases weekly, monthly or yearly in order to maintain the department's efficiency and competency.

4.3.4 Other functionalities

Severity will fall into categories such as:

- ☐ **Emergency**: Complete System Failure – the system does not handle processes properly and a manual intervention is needed to restore the system. Major disturbance – disturbance in the system functionality, that is the function does not work or is seriously affected.
- ☐ **High**: Major fault or disturbance affecting a specific area of functionality, but not the whole system. Major problems or disturbances that require immediate action such as restarts or reload. Failure affecting to the connection to the database. System crashes or hangs repeatedly. Critical function not available.
- ☐ **Medium**: Unable to perform non-critical functions, unable to process certain requests, questions regarding operations of the application
- ☐ **Low**: General questions regarding application (example: How to use?). Configuration consultancy.

With regards to the severity, the FLS will solve problems categorized under severity medium and low, should the problem is still unable to be solved than it will be escalated to SLS. The SLS will be assigned straight away to problems categorized as High and Emergency.

Next was to suggest the categorization of the problem itself. Problem description will be sub-divided into three categories:

- ■ Problem: default value for the requests
- ■ Consultation: Questions or new requests from the users.
- ■ Internal: Not requested by user but found in the application internally by any of the group members and need to be categorized as a maintenance activity.

All in all, for the functionalities suggested above, they all yield more than 70% of "Yes" votes.

4.3.5 User Interface

Each interface is briefly explained via Appendix A on how it works and respondents were asked if they agreed to the suggested interface. The interface begins with the FLS logging in to the tool once there is a support call. The FLS creates a request by inserting the user's name in a text field with a search button next to the text field. Once user's name is typed and search button is clicked, a pop up window will show the contact details of the user should he/she has logged any problem before. Click edit and the contact name and phone number of the user will be filled in automatically in the new request form.

Next the respondents agree that if the user does not exist in the system, the FLS call taker will create the new user's details. When creating the new customer's request, it is mandatory to fill in information such as: name, department name, office phone number, mobile number, email, title of the request, severity and problem description. For this suggestion, all developers agree to have this feature in the tool. The non-developers also agree by

82.4%. The respondents also agreed that should the request come via email, and there is an attachment, the FLS should attach the file in the system as part of the problem description. As common and viable this may seem to our assumption, all developers do agree to this however there are 17.6% of non-developers who disagree to this feature. This may be due to assuming that the user's attachment might not be useful or irrelevant to the problem. Nevertheless, users may also be knowledgeable in the sense that they do provide the correct picture of the problem indicated and that can further assist the maintainers in understanding the problem.

The FLS then selects an option button that says Analyze or Escalate. If the FLS chooses to "analyze" and click register, the problem is now registered in the system and assigned to him/her as a support work. If the FLS chooses "Escalate" and click register the problem is now assigned to the team leader of the SLS team.

Once register is clicked, there is a timer attached to the request to measure how long it takes for the specific person assigned to the job takes to solve the problem. Each group member will have their own job task list page in the tool to view their current request as well as list of previously solved request. Once they click on the new request, there is a status list box that allows the person to choose either New, Analysis, Pending answer approval or Closed. Once they choose analysis, then work is being done ether by providing solution through his knowledge or experience or search through a knowledge base on any related or previously solved similar problems.

If the solution is available in the knowledge base then the FLS will provide the remedy or carry out the restoration. After completing the restoration or correction then the status has to be changed to Pending answer approval. Once this status is chosen the solution text box will be visible for the support handler to fill in the solution to be updated to the knowledge base. It is interesting to note that for all the functionality mentioned above, all of the respondents agree to have the functionalities included in the tool as the result portrays 100% agreeing to each of the functionality.

Once an official answer is written via email to the user, the support handler will have to wait for the user to accept or reject the solution. If the solution is accepted then the support handler will change the status of the request to "closed". This will automatically stop the timer and logs the time end of the support process. If the solution is rejected then the cycle goes back to the user adding additional information regarding the problem and status changes back to analysis. For the functionality of writing a formal answer to the user via email, not all developers and non-developers agree to the functionality. Some fear that the answers could be too technical that the users may not appreciate or understand.

The next portion is to discuss the interface for the team leader and SLS. This will take place when there is an escalation from the FLS. The request first comes to the team leader of the development team, and according to team leader's experience the problem will be assigned to any of the team members. The team leader shall click on the list of problems escalated, reads the problem and click on an assigned list box which consists of the SLS team member's names. For this function, 95% of developers agree to implement this in the tool. This is further supported by 76.5% of non-developers agreeing to the same functionality. Once the SLS team member logs in, a new request is found in the inbox. The process is similar to the FLS activity that is the SLS will change the status to analysis and start analyzing. However, in the developer's status list box there is an extra item, which is "escalation to 3rd party". 3rd party would be problems escalated pertaining to hardware or device failure. The team member will start by searching the knowledge base if the similar solution is available; if it is not then the solution will be based on their knowledge and experience. The SLS does the remedy and restoration as well as updating the knowledge base. With respect to the functionality mentioned above, all respondents agreed to incorporate the functionality in the tool.

Once done, the SLS member will provide the answers and change the status to close. The SLS submits the solution which will then also update the status in the team leader's inbox. It is now the team leader's responsibility to check the status and once the status is at pending for user's approval, the team leader submits the request back to the originating FLS. The FLS then communicates with the user via email on the solution and waits for approval. The functionality on providing answers and change the status yields 100% "yes" by the respondents. However, the functionality on having the team leader to check the status , approve and respond back to the originating FLS who escalated the problem yields 50% of developers to agree and 50% to disagree.

This is probably due to some comments from the developers, that the team leader might ignore the status and causes late approval hence making the developers look bad from the user's perspective. Some developers feel that they should be made able to communicate directly to the user.

Moving on to the next feature, the team leader and manager is also able to view how many requests has been solved by the team members, weekly and monthly as well as the time taken to solve each request. Finally, the next module in the tool would be the documentation module in which the maintainer is prompted with two options:

- View the graphical functional flow of the supported application
- Search for the variable description

When the maintainer chooses to view the graphical functional flow of the supported application, the maintainer is able to see the functionality of the application and how it relates to one another. Upon clicking on the functions, a description on what the function does will be explained at the bottom of the window.

There is also a button should the maintainer would like to view the source code of the selected function. If the maintainer chooses to search for the variable description, he/she may type in the variable name or chooses the variable from a list that displays all the available variables used in the application. Once the variable is selected then a description on the variable is provided such as: usage in which function, data type, length and some description on the effects that will take place on the application should the variable is changed / modified.

5. CONCLUSION AND FUTURE WORK

Based on the findings the tool is seen as helpful and useful for the developers or maintainers of a university environment to assist them in their daily job activity. In addition to that, the suggested tool is also found to be especially useful for the managers to view and ensure quality within his team, through the time taken to solve each request. This can be supported by the functionality of having the timer measurement and viewing the solution solved. From the results it is also noted that the respondents agree to a proper process and line drawn between the job activity of a support engineer and developer.

From the survey conducted, there were a few relevant and strong comments that can be taken into consideration for future work in developing the tool. The respondents feel that it will be good if the tool incorporates a fixed service level agreement on what is the ideal time to fix each category of the problem according to its severity. For example, for the Emergency level, the maintainer is notified that it should be solved within 12 hours to 1 day, high within 2 days, medium within 5 days, and low within 7 days. This will draw the importance of the problem and allow the user to also know that they should get a solution within certain stated amount of time.

Next comment is on the pros and cons of having user agreeing to the solution, the respondents feel that some users are kind enough to close the case, and some are not. Therefore, suggestion is made so that the request is closed as soon as the FLS or SLS are finished with providing the solution. Another important comment is on having the Change Management on new requests to have a module by itself and not included in the problem sub-category.

All in all, the tool is needed as it could expedite the understanding of the application system, solving problems/troubleshooting and allow users to be part of the maintenance activity especially with important functionalities like request status, team leader and user’s approval of the solution, documentation module and knowledge base.

6. References

[1] Abran, Nguyenkin. 1991. Perspectives on Legacy SystemReengineering. www.sei.cmu.edu/reengineering/lsysree.pdf

[2] Bennet P. Lientz. 1983. Issues in Software Maintenance. Computing Survey, Vol 15.

[3] Dean Jin, 2005. Design Issues for Software Analysis and Maintenance Tools, 13th IEEE International Workshop on Software Technology and Engineering Practice (STEP'05).

[4] Ernst Lutz. 1993. The Knowledge Base Maintenance Assistant. IEEE

[5] Frank Niessink. 2000. Perspectives on Improving Software Maintenance. Software Engineering Research Center, Netherlands

[6] Harry M. Sneed, Peter Brossler. 2003. Critical Success Factor in Software Maintenance A Case Study. Proceedings of the International Conference on Software Maintenance.

[7] Hsiang-Jui Kung. 1998. Software Maintenance Life Cycle Model. IEEE

[8] Jane E. Huffman, Clifford G. Burgess. 1988. Partially Automated In-Line Documentation (PAID) Design and Implementation of a Software Maintenance Tool. IEEE

[9] Janice Singer. 1998. Practices of Software Maintenance. Proceedings of the International Conference on Software Maintenance, IEEE Computer Society

[10] Kagan Erdil,Emily Finn, Kevin Keating, Jay Meattle, Sunyoung Park, Deborah Yoon. 2003. Software Maintenance As Part of the Software Life Cycle. Comp180: Software Engineering Project.

[11] Kairulanuar Abdul Kadir. Analyst, Network Engineering Support Group. Ericsson Malaysia.

[12] Kathleen Brade, Mark Guzdial, Mark Steckel, Elliot Soloway. 1992. Whorf: A Visualization Tool for Software Maintenance. IEEE

[13] Keith Jones, Stephen Collis. 1995. Computerized Maintenance Management Systems. Property Management Journal.

[14] Kleiber D. de Sousa, Nicholas Anquetil, Kathia M. de Oliveira. 2004. Learning Software Maintenance Organization. Springer-Verlag Berlin Heidelberg.

[15] Massimo Felici.1997. Software Maintenance and Evaluation. Lecture Notes of SEOC1.

[16] Oscar M.Rodrfguez, Ana I. Martinez, Favela. 2004. Understanding and Supporting Knowledge Flows in a Community of Software Developers. Springer-Verlag Berlin Heidelberg.

[17] Robert L. Glass.1999. The “maintenance-first” software era. Journal of Systems and Software.

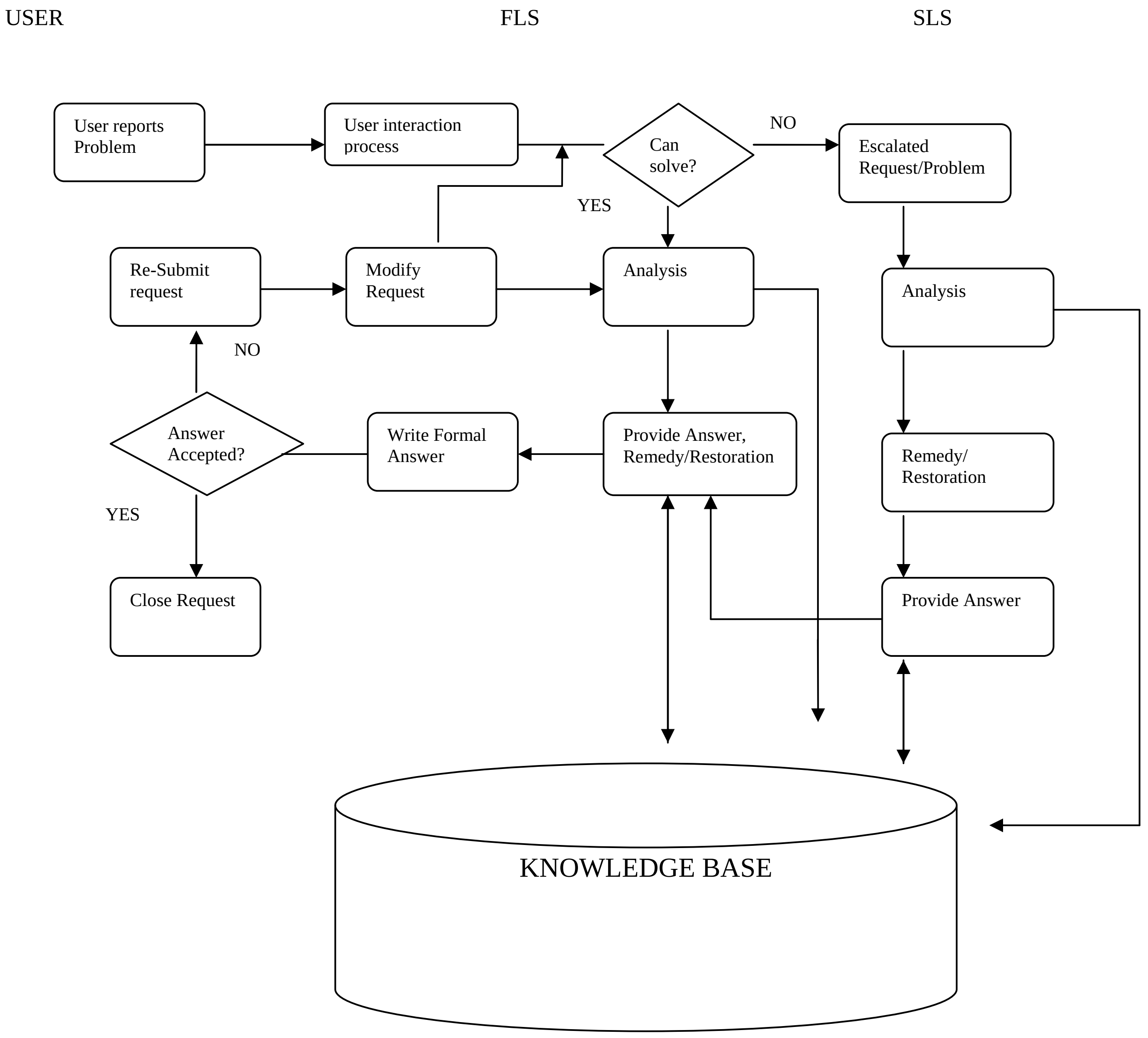
[18] Stephen W.L Yip. 1995. Software Maintenance in Hong Kong. IEEE

[19] Takang & Grubb. 1996. Software Maintenance: Concepts and Practice. hepguru.com/maintenance/draft.php

[20] Timothy C. Lethbridge, Janice Singer, 1998. Understanding Software Maintenance Tools: Some Empirical Research. IEEE Workshop on Empirical Studies of Software Maintenance

[21] Zelijka Car and Branko Mikac. 2002. A Method for Modeling and Evaluating Software Maintenance Process Performance. Sixth European Conference on Software Maintenance and Reengineering.

7. Appendix A: Process Flow of the Suggested Maintenance Tool


USER
FLS
SLS
User reports Problem
User interaction process
Can solve?
NO
YES
Escalated Request/Problem
Re-Submit request
Modify Request
Analysis
Analysis
NO
Answer Accepted?
Write Formal Answer
Provide Answer, Remedy/Restoration
Remedy/ Restoration
YES
Close Request
Provide Answer
KNOWLEDGE BASE