International Journal of Computer Applications (0975 – 8887)
Volume 156 – No 4, December 2016

Review of Software Maintenance Problems and Proposed Solutions in IT consulting firms in Mauritius

Sokappadu V. S. B. N
Dept. of Computer Science and Engineering
University of Mauritius

Mattapullut Gopaul S. D.
Dept. of Computer Science and Engineering
University of Mauritius

Rughoobur Paavan
Dept. of Computer Science and Engineering
University of Mauritius

Ramdoo Vimla Devi
Dept. of IT
Charles Telfair Institute
Mauritius

ABSTRACT
Software maintenance is often a challenging and hectic process for both software engineers and IT consultancy firms. Software maintenance is considered as the longest phase in software's lifecycle as it starts as soon as the software is deployed at the client-side and ends till the software goes obsolete. This paper sheds light on the various problems and challenges encountered by IT consulting firms in Mauritius during their software maintenance phase. A cause-analysis diagram has been used to illustrate the various problems along with their root causes. The paper then proposes some counteracting plans and solutions in order to alleviate those problems.

General Terms
Software maintenance problems, IT consulting firms in Mauritius, cause-analysis of software maintenance and solutions to software maintenance problems

Keywords
Software maintenance problems in Mauritius, solutions to software maintenance

1. INTRODUCTION
IT consulting firms are often faced with several problems during the maintenance phase where software developed becomes increasingly difficult to maintain and presents a set of new challenges, such as obsolescence and leaving staff. This paper is an advancement of a previous paper, namely "A Review on Software Maintenance Issues and How to Reduce Maintenance Efforts" [1] and is going to focus on the causes of maintainability problems in IT consulting firms of Mauritius.

2. SOFTWARE MAINTENANCE
Software maintenance process is very dense and usually comprises of more than half of the development process [15]. Software maintenance is the modification of a software product after delivery which includes correcting faults and failures [15]. Software Maintenance typically occurs as the last phase in most models (Refer to Figure 1 for the traditional Waterfall Model).

2.1 IT Consulting Firms
The Software Industry comprises a large segment of companies which come under the IT consulting umbrella. These companies enable organizations in achieving their business objectives using IT systems & often deploying software solutions which are custom-built according to the requirements of the client, following the maintenance phase. Last decade has witnessed the boom of IT consulting firms emerging and establishing themselves in Mauritius with both local companies such as The State Informatics Ltd and multinational companies such as Accenture (Delivery Centres). As such, much of the software projects ongoing in Mauritius follow the IT consulting model whereby software engineers develop and/or maintain solutions for their respective clients.

2.2 Maintenance in IT Consulting Firms
The maintenance phase in IT consulting firms start after a project has gone live, that is the project has already been deployed. Firms have different policies regarding the bearing of the costs of subsequent changes. Some firms provide a warranty period during which costs are maintained free of charge such as the State Informatics Limited in Mauritius charges a yearly fee for maintenance post release.

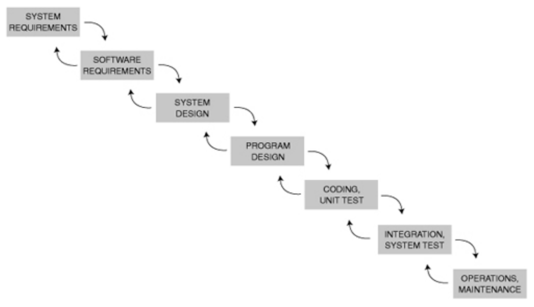

Fig 1. Waterfall Model (Boehm and Royce © 1970 IEEE)

2.3 Problems during Maintenance Phase
Worldwide, IT consulting firms are faced with a number of challenges and this is applicable in the Mauritian context as well. For instance, the problems are compounded in software system that are deployed overseas, and the maintenance engineers are in Mauritius (e.g. Accenture Delivery Centre in Mauritius where many projects are of the type "Maintenance Outsourcing"). In addition to the usual common causes of maintenance problems, factors such as time-zone differences, language barriers and local factors add on to the existing list.

3. COMMON CAUSES OF MAINTENANCE PROBLEMS
1. Technical Problems

Program/code comprehension: Software maintenance in IT consulting firms mainly includes changes that are of types adaptive and preventive in software systems. In order to make any change happen correctly and easily, the system attributes need to be understood by the maintenance engineers. Program/code comprehension is necessary for any change and it has been reported that this phase consumes more than half of maintenance resources available [2].

2. **Change impact analysis:** It is the process by which maintenance engineers assess the consequences and impact of each change on the components of the system [1]. This process leads to determining the overall cost of carrying out the change along with its feasibility issues. However in IT consulting firms, this analysis process is difficult to put into practice since the overall impact and coupling of system components is unknown or partly knowledgeable to maintenance engineers**.**

3. **Change implementation and propagation**: After impact analysis and feasibility study in the maintenance phase, the required change can be implemented on a specific component of the system. However, this change can have a regression effect on other components of the system that are interlinked, hence causing malfunctions elsewhere in the system. Thus, this may trigger additional changes to be applied to interlinked components, causing a chain reaction to occur. This chain reaction is called change propagation which may cause the system to be inconsistent to a certain extent if implementation of changes and/or testing are incomplete [2].

4. **Regression testing:** During maintenance, since changes are done in components of the system, regression testing becomes crucial. The purpose of regression testing is to ensure that changes made to the software, such as adding new features or modifying existing features, have not adversely affected other existing features of the software. Regression testing is usually performed by running some, or all, of the test cases created to test modifications made in the software. To ensure that all programs behave as expected, tests generated at earlier stages needs to be re-run.

However, as a program evolves, the regression tests set grow larger since old tests are also included, and hence the cost of regression testing increases. Due to pressure of time and budget constraints, repeating all previous test cases after each minor software revision or patch is often impossible and is not carried out in current practices [4].

5. **Database size:** During maintenance, engineers may as well make modifications to the structure and/or data of the database interconnected to the software system undergoing maintenance. In case whereby the database is of consequent size, repetitively applying these changes can be very risky and tedious since the maintenance engineers may not likely be trained to use contents and handle large databases [3].

6. **Product/system quality: T**he existing quality of the system can be a maintenance problem. The problems mainly are quality of programming and design issues such as non-compliance to standards.

7. **System age and obsolescence:** Older systems can pose several problems in maintenance process due to lack of maintenance teams' knowledge, product quality and programmer time. Older systems may also require frequent hardware and software changes as compared to recently deployed software systems [3].

8. **Operating environment:** These are the hardware, software reliability, software failure, data integrity and documentation of the software system. A large system is more error prone since a larger database requires a greater amount of change in data and files, as well as a greater need for hardware and software upgrades. With technological advances, old operating environments are more complex to maintain [3].

9. **System size and complexity: They are** measured by the number of program modules and the number of source statements contained in the system [1]. Hence maintenance in large system can be very complex and tedious in cases where the software system is highly complex.

10. **. Legacy software**: A legacy system is described as a software system based on outdated technologies and preceding generation computer languages, which are still in service and very critical to the daily organizational operations. There are a lot of such systems still in use in IT consulting firms and handling such systems includes many challenges not limited only to high capital investment in the initial system development but also inadequate maintenance documentations and lack of software supporting tools. A legacy software often requires many bug fixing, modifications, and updates along its life cycle. As the gap between the updated codes and outdated documentation broadens, maintenance becomes increasingly difficult in terms of effort and cost.

11. **Types of frameworks and building blocks used**: The type of framework used in the development of a software can cause an issue in the maintenance of the software in the future. One of the most common cause of maintenance problem software firms have to face is that they are unable to find enough support or skilled resources to maintain their software due to poor decision on framework choice at the moment the software was developed.

12. **Restructuring for change**: During the maintenance process, changes may require restructuring to be done, since the architecture in place do not support the changes.

a. Managerial (Non-Technical) Problems

1. Programmer's availability: In IT consulting firms, the maintenance team has to work on several projects simultaneously. Hence the maintenance programmers have the time restriction on each request for change in each project.

2. User's demands and expectations: Software user knowledge contribute to the factors of maintenance problems in IT consulting firms. One case is that users can have limited knowledge of the system due to lack of training and can have high expectations from the maintenance team through their change demands which can be out of the software domain or out of the boundaries of the maintenance team. Another problem from user expectation is that they expect their changes to be implemented and deployed in a short lapse of time, and this may not be feasible for the already overburden maintenance team.

3. Staff size: It describes the number of people who are engaged in the development process of software under application development. After delivering the project, customers surely need some changes and then the software team which is also engaged in the other development projects, is assigned the additional work. This increases the efforts of the development staff. Software team cannot assign the work of maintenance to other free developer because the other developers are completely unaware of the delivered project, which is under maintenance. If software organisation assigns the maintenance work to new developers or programmers then the organisation needs to provide training to the programmer, which leads to the increment in time, cost and efforts for the maintenance [1].

4. Staff turnover: Maintenance problem includes staff turnover as well. In case there is a high staff turnover in the maintenance team, the new replacement members of maintenance team will not have the sufficient knowledge in dealing with changes in software systems [1]. Hence more time should be dedicated in the training of the new team members, thereby increasing maintenance cost alongside.

5. Maintenance budget: It is the amount of money earmarked by a company to be spent on maintaining its current system. A limited budget signifies deficient maintenance and inadequate testing which may result in new bugs cropping up in the system.

6. Documentation quality: The documentation available during the maintenance process in IT consulting firms are more than often of poor quality and non-compliant to standards. It can also happen to have various versions of the same feature documented in different documents. This in turn results to high costs in terms of maintenance effort in order to find and correct faults in the system.

7. Development experience of maintenance staff: More often in IT consulting firms, the maintenance team and the development team are two separate teams. Due to inadequate training and less interactions with the software during its implementation phase, the maintenance team has limited knowledge on the system. It can also happen that the maintenance process is outsourced to a maintenance team from another IT consulting firm. This leads to limited commitment from the development team during the maintenance process.

8. Project management issues and failures: Poor project management and development can also lead to poor software maintenance. Very often, stakeholders and management are reluctant in allocating budget for maintenance of a project which they were not involved in or encountered several problems during its development phase.

9. Lack of skilled labour: Lack of skilled resources is another major cause of poor or prolonged software maintenance. It has been found that firms are very resource dependent for some particular software maintenance as it will be very costly to train new resources for the maintenance of these software.

4. CAUSAL ANALYSIS

Carrying out a causal analysis process reveals the cause-and-effect relationship among all the enumerated factors discussed in Section 3 (See Figure 2 for the cause-analysis diagram).

The causes encased in rectangles refer to normal causes. Asking the cause(s) ("Why?" question) by inference leads to other causes with the final root causes enlisted in the hexagons.

1. Program comprehension problems, that is the difficulty in understanding the software to be maintained can be caused by a set of factors including poor product quality (*Product quality factor*), complex and less known frameworks used (*Types of frameworks and building blocks used)*. For example, program comprehension problem may arise if maintenance engineers are themselves unable to understand the inner workings of the software being maintained.

2. Effort in change impact analysis: *Program comprehension problem*s can in turn cause more time taken to estimate the impact of changes, increasing the **effort in change impact analysis.**

3. Difficulties in change implementation can be caused by poor product quality problems (*Product quality factor*), high program complexity (*System size and complexity factor*) and a small maintenance budget (*Maintenance budget factor)*. While a products of inferior quality or high complexity may result in high effort (even after the software has been well understood), small budgets allocated for the project impose time constraints thereby reducing time for quality activities.

4. Over-regression can be a problem where too much regression testing is done to ensure the software is still stable after a modification, and is often the result of poor program comprehension (P*rogram comprehension factor).* The impact of a change is unknown and though it may not impact several other modules or pathways, they are all tested for fear of introducing breaking code-changes.

5. Limited programmer time availability is another problem which can delay maintenance requests or lead to inferior quality. This is typical of several IT consulting firms where maintenance engineers are Software Developers on other projects which are in the software development phase. Common causes for these are *Staff Size, Small Maintenance Budgets and Project Management issues.*

6. Limited user knowledge is a problem caused commonly by poor training of the users and poor documentation manuals (*Poor product quality factor).*

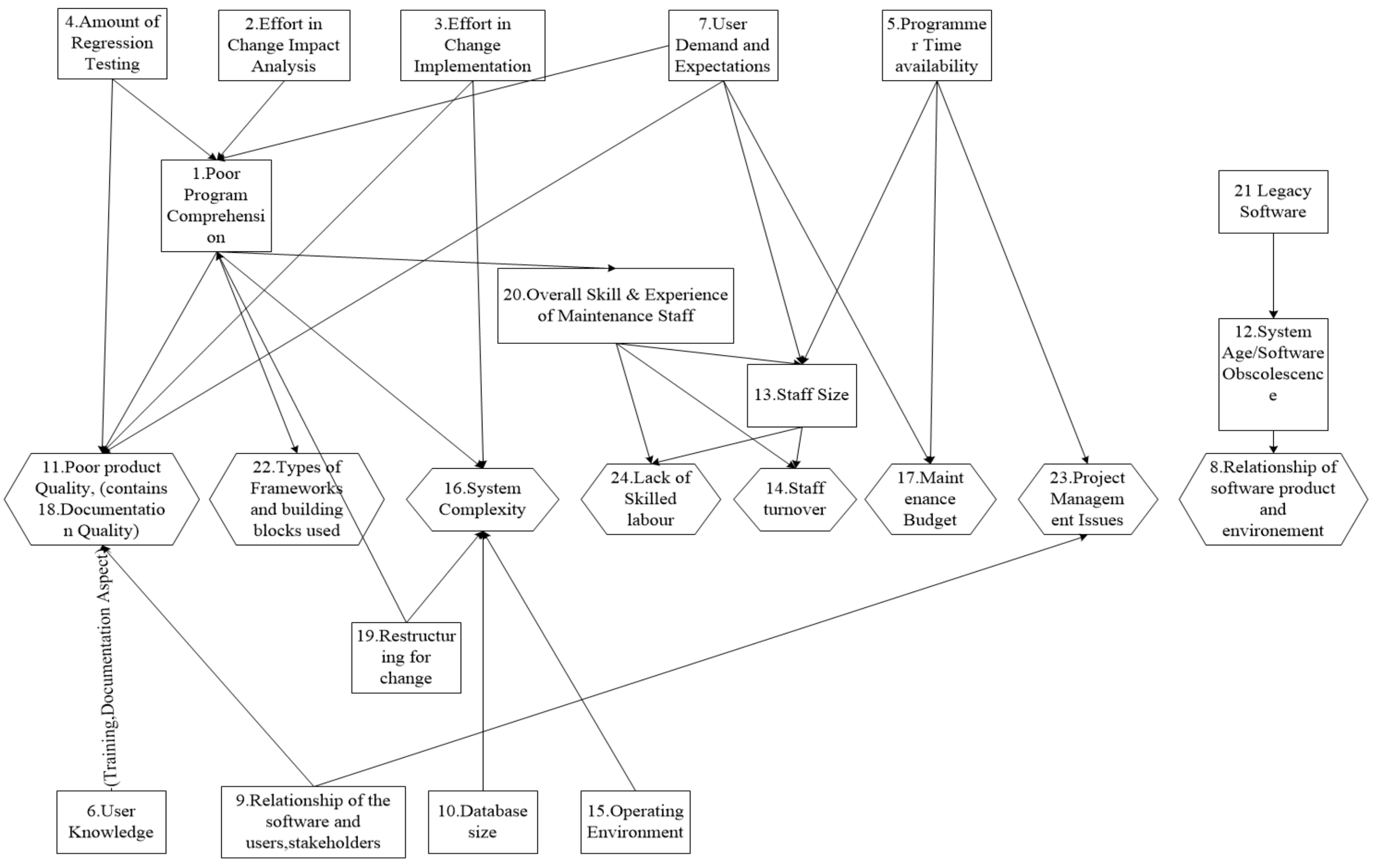
4.Amount of Regression Testing
2.Effort in Change Impact Analysis
3.Effort in Change Implementation
7.User Demand and Expectations
5.Programmer Time availability
1.Poor Program Comprehension
21 Legacy Software
20.Overall Skill & Experience of Maintenance Staff
12.System Age/Software Obscolescence
13.Staff Size
11.Poor product Quality, (contains 18.Documentation Quality)
22.Types of Frameworks and building blocks used
16.System Complexity
24.Lack of Skilled labour
14.Staff turnover
17.Maintenance Budget
23.Project Management Issues
8.Relationship of software product and environement
(Training,Documentation Aspect)
19.Restructuring for change
6.User Knowledge
9.Relationship of the software and users,stakeholders
10.Database size
15.Operating Environment

Fig 2. Cause-Analysis Diagram for Software Maintenance Problems in IT Consulting firms

7. User demands and expectations may seem relatively very high in proportion to what the maintenance project can offer due to reasons such as *poor program comprehension, poor product quality, reduced staff size and reduced maintenance budget.* For instance, changes which seem relatively simple at a functional level can be extremely difficult to change within the software due to high complexity of the software or unavailability of adequate staff to service the request. As a result, for example, a change request taking a 1-day man effort may be stretched beyond a week to get serviced.

8. The relationship of the software product and the environment has been identified as a root cause in the graph (see Figure 1.) When the software has an inherently non-adaptive relationship with the operating environment or the contemporary context in which it is being evaluated, this has a detrimental effect on its use and acceptance. For example, an ERP software developed by X company but customized and sold by ITZ company and which is no longer supported by X company has lower chances of being successfully maintained in the long run.

9. The relationship of software product and the stakeholders has been identified as an effect of 2 causes in the graph (see Figure 1.) A product of inferior quality elicits poor response from its users. A wrong managerial approach to enforce the use of a software among the users can demotivate its use. Engineers who have had a negative experience developing the software, despite the product itself being of good quality might be less willing to undertake work regarding such software. These both factors affect the perception and the type of relationship the stakeholders have with the product and may affect the willingness to ensure proper maintenance.

10. A large database schema can be a result of a very complex product (*System size and Complexity*). A large database (in terms of definition rather than the size of data stored) is the effect of a highly complex system. An application connecting to 2 tables may be easier to maintain and manage rather than one connecting to hundreds of tables. A foreign key constraint change for example, can break applications and the impact is higher if the extent of interconnectedness is greater.

11. Product quality has been identified as a root cause as it can be quantified using metrics or can be simply an opinion based on some evaluation criteria. For example, a software system that does not have a Help System can be perceived to be a product of lower quality. A software that has a very high crash recovery time is similarly of inferior quality. McCall's Model and FURPS+ are among several models can be used to define the quality of a product.

12. System age and software obsolescence occurs due to the fact that a software is too tightly coupled with its environment and is unable to adapt to new environments (*Relationship of the software with the environment factor).* For example a software whose lowest level instructions are heavily dependent on the types of registers and underlying hardware used, such software may be difficult to maintain once the underlying hardware is upgraded to a newer version.

13. Limited staff size is an effect of *unavailability of skilled labour* or a high *staff turnover* as well. Limited staff can create a "lower transparency" effect on the servicing of maintenance requests. For example, a service request may be estimated to be an hour in terms of budget. While one would expect the request to be service in a short period of time, the request itself may finally be serviced after several days, owing to the fact that the personnel are busy servicing other requests in the queue. Such incidents can contribute to a negative client experience.

14. Staff turnover has been identified as a root cause as it essentially contributes to a domino effect in software projects contributing to an overall poor maintenance effort. Staff turnover is itself an effect of several issues such as poor management practices and retention effort, job market and so on. A high staff turnover implies that considerable time is being spent in training new employees. Despite external factors, smart companies should be able to mitigate this problem or take steps to maintain a consistently skilled pool of individuals at all times.

15. Unfavourable operating environment can be a product of a highly complex system (*System size and complexity)*. System can be coupled with a lot of software, hardware components and might be heavily dependent on other systems. Alternatively it can be a part (subsystem) of a bigger system. In such cases, changes may require a significant amount of regression testing or create unforeseen changes within the system as a whole.

16. System size and complexity has been identified as a root cause as system complexity can be quantitatively measured using metric such as McCabe's Cyclomatic Complexity. For instance a software have only one flow of execution is much simpler to change and maintain as compared to a software with hundreds of permutations of flows of execution where rework and testing may take higher time. A highly complex system is problematic to maintain as it is more difficult to comprehend.

17. Small maintenance budget has been identified as a root cause since tight budgets are usually being allocated for maintenance phases by the IT consulting firms. This arises due to an erroneous assumption that the maintenance phase is the least important phase of the Software Development Life Cycle (SDLC) or that very little effort is needed to maintain already developed and deployed software projects.

18. Documentation quality is not a standalone cause on its own. Though identified in previous research [1] as a separate cause, it is a subset of the attributes of a product of inferior quality (*Product quality factor).*

19. Restructuring for change becomes difficult if the system is of complex nature (*System size and complexity factor*) or the system is of inferior quality (*Poor product quality).* A highly complex software system is more resistant to changes as there is an inherent tendency not to disturb systems which are already working well. It is also difficult to introduce changes into a system that is of poor quality. For instance a poorly documented and commented program required more time for comprehension in order to bring changes to the codes. (*Program comprehension)*

20. The limited overall skill or development experience of maintenance staff may seem to be very low due to reasons such as low *Staff size* and high *Staff turnover.* As such the required knowledge to service certain requests raised by the client may even be inexistent. A maintenance project servicing a highly critical application but staffed only by 2 inexperienced developers with a high turnover rate (1 per month) may have high chances of failure than a similar project with at least 1 experienced developer with a lower turnover rate.

21. Legacy software issues are caused by *System age/software obsolescence factors.* Some legacy software problems caused by aging software include adaptability problems, dependencies on older technologies and infrastructure, and even a growing lack of professionals undertaking training in the legacy technology using which the software was conceived.

22. Types of frameworks and building blocks used has been identified as a root cause as some frameworks may be much lesser known than the others or less widely used. In turn resources such as learning material, questions and answers regarding bugs and troubleshooting may be not be available.

23. Project management issues has been identified as a root cause as they include a myriad of issues ranging from sensitive people related issues to hard technical problems arising from maintaining the software. This root cause essentially enlists all the failures and shortcomings on the project manager's and higher management's side which pose problems in software maintenance projects. One project management issue could be the very low priority and as a result, negligence towards software maintenance projects which result in effects, such as resources constantly moving to higher priority projects, or the lack of responsiveness to problems escalated to the manager. Software maintenance projects often suffer as a result since blocking points and choke points are delayed by the sluggishness with which such projects are inherently handled.

24. Lack of skilled labour has been identified as a root cause as it gives rise to several human resource and staffing related issues such as lack of skills within the project team or a small staff size. Lack of skilled labour is a management, more specifically a human resource management problem along with contemporary issues prevailing in the country for example political stability, economic growth, and education policies which affect the job market. Such problems are often alleviated through the hiring of resources from overseas. In the case of Mauritius, countries such as India, Madagascar and Cameroon remain good labour pools from which skilled professionals are brought. Skilled labour deficiency may also be attributed to poor retention policies, poor working conditions and factors coming from locally and from within the company.

5. CAUSES OF MAINTENANCE PROBLEMS SPECIFIC TO MAURITIUS

Software maintenance problems in IT consulting firms in Mauritius have other root causes other than the aforementioned common problems.

b. Geographical Location and Country-Specific Factors for Overseas Projects

These reasons are especially relevant for software projects which have foreign clients or have been outsource to Mauritius.

5.2.1 Time zone differences

Several IT consultancy firms, such as Accenture, work with clients abroad from a wide range of regions including Europe (France) and Canada. The time zone differences, although at times advantageous, may be disastrous in other scenarios. For example if a critical bug occurred in a France-based project at around 17 00 at the same time in Mauritius (19 00 or 20 00 based on summer-time daylight savings) the maintenance team would have been functioning with minimal resources since it is past the normal working hours locally. The Mauritian team may therefore not be prepared to respond to a critical downtime as it should in normal working hours.

5.2.2 Geographical barriers

As maintenance consultants, it is much easier to intervene at the client site physically to fix a bug or for normal maintenance rather than having to do it remotely, often with the assistance of technicians or infrastructure engineers abroad over a TCP/IP network connection.

5.2.3 Language Barriers

Communication problems may arise due to differences in dialect and accent. Although two individuals from different countries are communicating and may be well-versed in the same language, misinterpretations may crop up in documentation or even verbal communication, leading to even further delays.

5.2.4 Organization Culture

Day-to-day habits may introduce problems and time lags in response time or treatment time of issues. For example, French organizations and Mauritian organizations may have different lunch cultures and durations. Mauritius may have public holidays because of Hindu festivities while Canada does not.

6. PROPOSED SOLUTION PLAN FOR PROJECTS IN MAINTENANCE PHASE

c. Maintenance Issues and Quality

In an ideal scenario with perfect project management processes in place, the software produced should be bug-free. Maintenance issues are therefore reflective of a software's quality especially corrective maintenance.

d. Checklist Approach

The following table enumerates solutions for different types of problems in the Software Maintenance phase (See Table 1). The check-list employed helps to clarify which activities should be carried out and by whom and lists solutions which can be employed for different types of problems identified. Individual solutions are elaborated below.

Table 1. Table captions should be placed above the table

Root Cause Addressed	Question	Solution	Type of Solution
Product Quality	Is Documentation right?	Re(Documentation)	Technical
	Is Unnecessary Code present?	Eliminate Dead Code Eliminate Cloned Code	Technical
	Is Code Buggy?	Eliminate Bugs	Technical
	Is Code ugly?	Refactoring	Technical
Product Quality, Complexity, Building blocks	Design Changes needed?	Restructuring	Technical
Product Quality, Complexity, Building blocks, Product/Env relationship	Is restructuring not enough?	Re-engineering	Technical
Human Resource problems	Do we have good enough people? Do we have enough good people?	Technical Training, Outsourcing, Recruitment	Managerial
	Are problems due to country-specific reasons?	Apply Project Management Practices.(Leadership), Training(Language)	Managerial

Without documentation, a programmer spends 21.5% time in understanding of code. With documentation, 12% of cost of maintenance could have been saved [1]. Maintenance cost can be reduced by (re)documentation using Computer Aided Software Engineering (CASE) tools in automated documentation [14]. LOWER CASE tools support the implementation and maintenance phases of the systems development life cycle [14].

CASE tools helps in:

- Substantial savings in resources required for software development,
- Shorter time to market,
- Substantial savings in resources for maintenance,
- Greater reuse due to increased standardization of the software systems, and
- Reduced generation of defects coupled with increased 'interactive' identification of defects during development [13].

The main component of CASE tools is the repository which stores all changes and information of the project from development to maintenance phase [14].

CASE documentation generator tools can also be used as it simplifies production of technical and user documentation and contains master templates to verify that documentation conforms to all stages of SDLC [14]

e. Dead Code Elimination

Dead code is a section in the source code of a program which is executed but whose result is never used in any other computation [11]. The execution of dead code wastes computation time and memory. Software systems may contain significant proportions of dead code which can go as high as 30 percent according to the Omnext white paper that was published on March 2010 namely "How to save on software maintenance costs". [16]

IT projects often contain dead code segments which been added by developers for future reference (for example showing how to perform a particular operation or use a particular function), future use (upon subsequent adding of functionalities), or for performing a simple rudimentary test (to see whether a variable has the correct value). Often this type of code, if uncommented, can lead to confusion as to why it was included – finding out the reasons for its presence can lead to the unnecessary waste of effort.

Dead code stripping/elimination/removal is usually a compiler optimization to remove code which does not affect program results. Dead code elimination removes unnecessary lines of code that reduces overall size and complexity of the software.

6.3.1 Duplicate/Cloned Code Elimination

Duplicate code refers to the repetitive occurrence of a sequence of code more than once in a program or in several programs belonging to the same entity. There is a limit to the amount of code that can be repeated beyond which is considered as a duplicate. The duplicate sequences are usually known as code clones or clones and the process of finding them code clone detection. Duplicate code may happen because of a number reasons:

- Copy-Paste programming or scrounging, that is where a segment of code is copied and pasted because "it works". This is typical of IT consulting firms which operate on tight budgetary constraints and schedules. Often there is a tendency to write code "that gets the work done" rather than optimized and readable code. Copy-pasted code may contain redundant operations which may not be needed for the program into which it is being copied.
- Rewriting same functions again: A programmer redefines a segment of code which already exists in another place either knowingly or unknowingly.

  - Generated codes, which is code generators may generate duplicate code because it is simpler. Project teams may adopt this method because of speed, despite the redundancy problem.
- Inappropriate code duplication may reduce maintenance costs [17].

Research shows that large software systems can contain from 10% to 25% of redundant code [1]. Removal of redundant code would make the software simpler to understand and thus would help to reduce the effort needed to learn and maintain the system.

6.3.2 Bug Detection and Elimination

Bugs in software are costly and difficult to find and fix. Techniques and tools have been developed for automatically finding bugs by analysing source code [1]. For instance the problem of syntax errors can be eliminated through the use of compiled languages rather than interpreted languages.

6.3.3 Software Refactoring

Refactoring is a preventive maintenance practice that intends to improve or refine software, thus slowing down its degradation. It is a continuous process that improves the non-functional attributes of the software without changing its external functionality. It contributes to improving the structure of a program, reduction of code complexity and increase of usability. Refactoring therefore directly improves program comprehension and reduces system complexity. Simpler program structure and more readable code are easier to maintain and consume less man-hours. Refactoring is however not a standalone practice but is constituted of practices discussed above such as Dead Code Elimination, Cloned Code Elimination and Bug Elimination.

6.3.4 Re-engineering

Software Re-engineering constitutes 2 processes namely reverse engineering and forward engineering. Reverse engineering can be defined as "the process of analysing a subject system to identify the system's components and their relationship and to create system in another form or at a higher level of abstraction [10]. Reverse engineering tools extract data, architectural, and procedural design information from an existing program [1]. The reengineering process uses this extracted information for analysis and redesign and tries to identify the components that have to be reused and the parts that need to be redesigned or replace. After the final design has been created in the reengineering process the forward engineering process takes over in implementing this design and finally testing the new modified system [12].

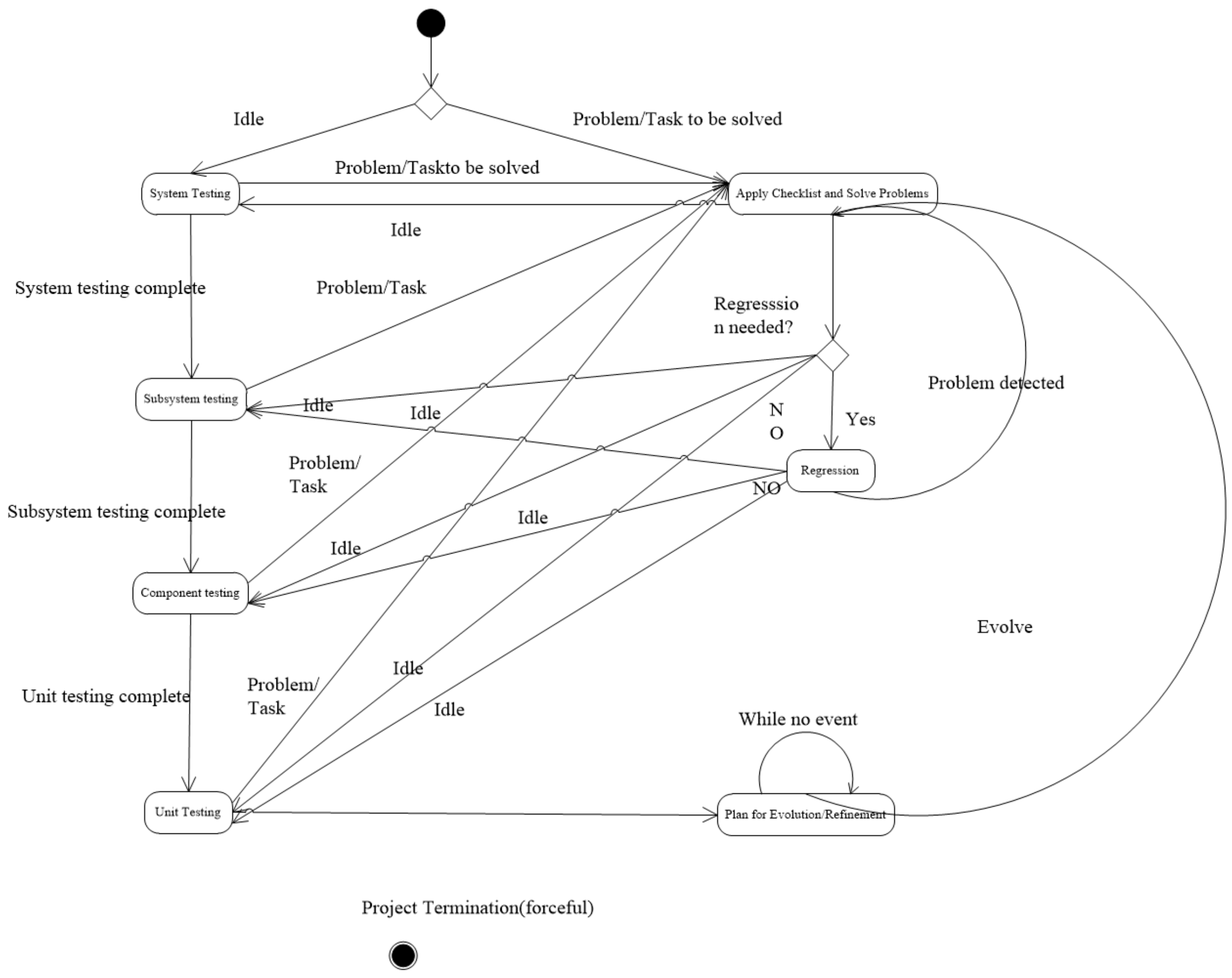

Fig 3. Statechart Showing Suggested process model for Maintenance Projects In IT Consulting Firms

f. Software Maintenance Flowchart

Although problems in software maintenance projects or any form of software projects can never be removed, a reduction in number of issues is possible. Figure 2 shows the following procedure that are being proposed for handling software maintenance projects.

The process model offers a mode of operation of software maintenance projects especially adequate for IT consulting firms in Mauritius with small dedicated teams performing only software maintenance. The model incorporates the resolution of problems while it also depicting how the project may use idle time for quality-related activities.

- A software maintenance project team may either be in 2 general types of states: Idle or Working. When a project has begun, the team may be idle and awaiting work. In such a scenario, the team may itself initiate a system testing phase to recover and anticipate defects and if possible fix them. The activities undertaken follow a shallow-to-deep approach. System testing is carried out to eliminate defects which are most likely to be caught through general use. Following system testing, subsystem testing activities are carried out, and after that unit testing-provided that the project allows the team to descend to such low levels of granularity.
- In the event the team receives work, for example in the form of an adaptive maintenance request, the request may be fulfilled. In such cases if the change is suspected to have impacted other areas within the system, a regression testing phase may be introduced. Otherwise, the team may resume its normal exploratory testing activities.
- The flowchart indicates that whenever a team has work it switches to the assigned tasks to be completed and in case it does not it switches to the completion of system testing. This guarantees that at no point the team is idle.
- After testing activities have been completed, and the team is scheduled to be idle under normal circumstances, the flowchart introduces a “Plan for Evolution/Refinement” phase during which the team may undertake the improvement of the software without external intervention. Again the checklist solution provided can be referred to for

applying software improvement practices such as re-engineering/re-factoring. Alternatively the team can undertake training sessions for improving the standard of quality provided.

The proposed flowchart therefore helps to maximize human resource usage for a software maintenance project and provides guidance regarding how a team should ideally operate.

g. Solutions Specific to Mauritius

1. Global team operation facilities – companies and project teams operating should put in place systems and project management tools which allow easy collaboration and communication within projects. For example, source code versioning tools can help track code changes. Putting into place reporting and status tools can help dissipate important information regarding project status immediately. Putting into place flexible timing systems, VPN access facilities, and encouraging high bandwidth communication are some measures which can reduce the disparity created due to time zone differences between Mauritius and abroad.

2. Overcoming labour shortage and language barriers through education – Skillset mismatch and shortage of labour can be overcome by conducting courses most relevant to contemporary IT industry. For example, language courses and coaching offered by Ministry of Education could help alleviate certain problems due to communication gaps. Encouraging tertiary institutions to offer B.Eng courses in IT-related streams such as Computer Science and Engineering could help produce significant labour to overcome labour shortage problem. Relaxing immigration laws could also help bring more labour to meet the skillset shortage gaps.

h. Further Recommendation

Some practices seem to offer better advantages than others, for instance including a member of the original development team in the maintenance team as a member of the actual software development team has first-hand knowledge of the software product and can be said to have more expertise than new maintenance engineers. The maintenance engineer can therefore easily resolve various problems encountered.

7. CONCLUSION

The different problems relating to software maintenance affecting IT consulting firms in Mauritius have been addressed in this paper along with an attempt to distinguish the relationship between the different factors. Solutions have been proposed in order to alleviate the different problems encountered. Nevertheless, after so many years, it is found that the software industry is still fighting with the same issues in software maintenance and sustainable solutions is yet to be found and implemented.

8. REFERENCES

[1] Kaur, U. and Singh, G (2015). A Review on Software Maintenance Issues and How to Reduce Maintenance Efforts. *International Journal of Computer Applications on* 118-1: 0975-8887.

[2] Bennett, K.H. and Rajlich, V.T., 2000, May. Software maintenance and evolution: a roadmap. In *Proceedings of the Conference on the Future of Software Engineering* (pp. 73-87). ACM.

[3] Palvia, P., Patula, A. and Nosek, J., 1995. Problems and issues in application software maintenance management. *Journal of Information Technology Management*, *6* (pp.17-28).

[4] Wong, W.E., Horgan, J.R., London, S. and Agrawal, H., 1997, November. A study of effective regression testing in practice. In *Software Reliability Engineering, 1997. Proceedings, The Eighth International Symposium* (pp. 264-274). IEEE.

[5] Sannella, M.J., 1994. Constraint satisfaction and debugging for interactive user interfaces (Doctoral dissertation, University of Washington).

[6] Forman, G., 2003. An extensive empirical study of feature selection metrics for text classification. *Journal of machine learning research* (pp.1289-1305).

[7] Leonard D. Brown, Hong Hua, and Chunyu Gao. 2003. A widget framework for augmented interaction in SCAPE. In *Proceedings of the 16th annual ACM symposium on User interface software and technology* (UIST '03). ACM, New York, NY, USA, 1-10.

[8] Yu, Y.T. and Lau, M.F., 2006. A comparison of MC/DC, MUMCUT and several other coverage criteria for logical decisions. *Journal of Systems and Software*, *79*(5), pp.577-590.

[9] Spector, A. Z. 1989. Achieving application requirements. In Distributed Systems, S. Mullender, Ed. Acm Press Frontier Series. *ACM Press*, New York, NY, 19-33.

[10] Canfora, G., Cimitile, A. and Lucarelli, P.B., 2000. Software maintenance.*Handbook of Software Engineering and Knowledge Engineering*, *1*, pp.91-120.

[11] Debray, S.K., Evans, W., Muth, R. and De Sutter, B., 2000. Compiler techniques for code compaction. *ACM Transactions on Programming languages and Systems (TOPLAS)*, *22*(2), pp.378-415.

[12] Kidambi, P.C. 2003. Maintenance Issues in Software Engineering. *Department of Computer Science Louisiana Tech University.*

[13] Galin. 2004. Case tools and their Effect on software quality in SQL from theory to implementation, eds Pearson Education limited, Chapter 13.

[14] Hoffer, JA, George JF & Valacich JS. 2005. Automated Tools for Systems Development in Modern Systems Analysis And Design,4th edn, Pearson/Prentice Hall, Appendix 2.

[15] Parul, DK. 2014. Challenges during Software product maintenance, *International Journal of Computer Science*, vol. 2 (3).

[16] Engelbertink, F.P. and Vogt, H.H., 2010. How to save on software maintenance costs. *Omnext White Paper.*

[17] Kapser, C. and Godfrey, M.W., 2006. Cloning considered harmful considered harmful, *13th Working Conference on Reverse Engineering* (pp. 19-28). IEEE.