Volume 7, Issue 3, March 2017 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Justification of Software Maintenance Costs

Nexhati Alija
International University of Struga,
Macedonia

DOI: 10.23956/ijarcsse/V7I2/01207

Abstract- ***The whole life cycle of a software solution might be partitioned into: 1) production phase, and 2) maintenance and support phase. However, all reliable calculations are telling that the most costly is the maintenance and support phase, as it counts approximately ninety percent of the cost. Common people, being unfamiliar with the length of the process, its various cost weight factors, and other maintain activities and expanses, usually think that the maintenance of a software application is unreasonably costly. If vendors and users want to have a greater business understanding and mutual trust regarding their maintenance charges and fees, then they really have to become knowledgeable of those lengthy processes, pondering factors, and all the maintenance's activities, that are concretely affecting their application maintenance costs/fees. Contrary to the software development phase, which is much highly profiled among general public, the maintenance phase itself does not receive the same level of attention and recognition. Organizations, in particularly have to change their misconception of software maintenance phase importance and costs.***

Having that in mind, and targeting readers who do not have clear picture about software maintenance, this paper work clarifies that:

- ***maintenance is an integral portion of a software's life-cycle which includes bug fixing as well many other various and complex labor intensive intellectual activities; and***
- ***maintenance costs, being entirely justified, increase the value and performances of the software itself, and therefore are to be considered as an investment.***

Keywords: Software Evolution, Software Maintenance, Software Maintenance Costs.

I. INTRODUCTION

Common people that are not part of the engineering community, due to being unfamiliar with the length of the process, its various cost weight factors, and other maintain activities and expanses, usually think that the maintenance of a software application is unreasonably costly. Usual misconception among such people is that vendors unjustifiably directly increase the maintenance fees and tightening application of current affluent contracts.

Most commonly, people think that maintenance fees are mainly for two types of services: firstly, for software development which should provide new functionalities, upgrades related to new regulations, and bug-fixing patches (paying for which they considers being unfair, as they already paid for acquiring a software which has no built-in defects/bugs); secondly, for maintenance which should cover phone and internet based user support in cases when they need usage help. Contrary, the software maintenance engineers have hard time to explain that not always it is the same person who develops and maintains a software solution, and therefore they usually face with poor insight (due to poor or no technical documentation) of the design and the source codes.

As a result, maintainers have to spent some time to understand all the interconnections and underling principles, in order to better understand how and where to intervene in the existing design and code prior to modify and to implement changes in accordance to users' needs or new regulation, envisioning affections to the overall software system. On the other hand, such widespread misconception is partially resulting from not fully correct understanding of the term "software", and without it the software maintenance would be quite impossible being properly understood.

If vendors and users want to have a greater business understanding and mutual trust regarding their maintenance charges and fees, then they have to become knowledgeable of those lengthy processes, pondering factors, and all the maintenance's activities, that are concretely affecting their application maintenance costs/fees. Because of this need, it would be very beneficiary in this paper, before stating some various definitions of this concept, firstly to provide definitions of terms such Software Evolution, Software Maintenance, Software Maintenance Costs.

- Term Evolution applies to a process of continuing modifications leading from a simpler towards more complex state.
- Term **Maintenance** applies to activities such as regular or occasional repairs and updates, in order to keep the effectiveness, efficiency and/or validity, required to prevent failures or decline.
- Term **Maintainability** signifies the level of easiness with which maintenance can be performed.
- Term **Software** means computer programs, documentation and operating procedures making a computer usefully performable.

Table I Example of typical Software Components:

Software Components	Examples		
Program	1 Source code		
	2 Object code		
Documentation	1 Analysis / specification:	(a)	Formal specification
		(b)	Context diagram
		(c)	Data flow diagrams
	2 Design:	(a)	Flowcharts
		(b)	Entity-relationship charts
	3 Implementation:	(a)	Source code listings
		(b)	Cross-reference listings
	4 Testing:	(a)	Test data
		(b)	Test results
Operating procedures	1 Instructions to set up and use the software system		
	2 Instructions on how to react to system failures		

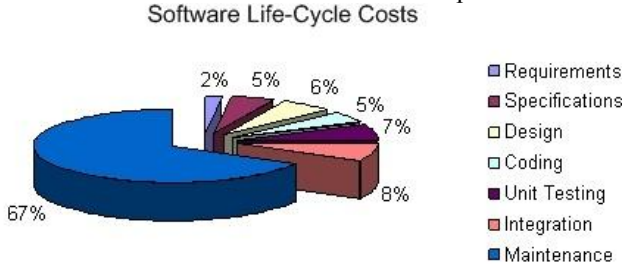
Although there are numerous definitions regarding Software Maintenance, the following are the mostly accepted ones. According to Institute of Electrical and Electronics Engineers (IEEE 1219), "Software maintenance is the process of modifying a software system or component after delivery, to correct faults, improve performances or other attributes, or adapt to a changed environment". According to ISO/IEC 12207 Standard for Life Cycle Processes, "The maintenance process contains the activities and tasks of the maintainer. This process is activated when a system undergoes modifications to code and associated documentation due toan error, a deficiency, a problem, or the need for animprovement or adaptation. The objective is to modify an existing system while preserving its integrity". According to the SWEBOK (Software Engineering Body of Knowledge), "The totality of activities required to provide cost-effective support to a software system. Activities are performed during the pre-delivery stage as well as the post-delivery stage. Pre-delivery activities include planning for post-delivery operations, supportability, and logistics determination. Post-delivery activities include software modification, training, and operating a help desk".

Because not always all the changes to the software are done all together at the same time and with the same reason / cause, it is recommendable to understand the nature of each particular change made to software, so that we can always distinguish between maintenance types.Distinguishing helps us to better understand the importance and implications of a maintenance to the overall system performance as well to the overall cost implied. Classifying maintenance conducts into category types evidences that the software maintenance done is more than fixing bugs. Lientz and Swanson[1)] classify maintenance into 3 types: Corrective; Adaptive; and Perfective.

All the changes done in order to overcome actual faults in the softwareare considered being Corrective maintenance. All the changes provoked by some mutation in the operating environment where the system performs, are considered being Adaptive maintenance. Changes which are initiated by user due to improve and/or ease functionalities are considered being Perfective maintenance. Worth of mentioning here is that according to **Pigoski[2)]**, the *Adaptive* and the *Perfective* category typesdue to their non-corrective nature, both being improvements, should be blendedand be referred as the *Enhancement type*. On the other hand, **IEEE[3)]** redefines the Lientz and Swanson[1)] categories of *Corrective*, *Adaptive*, and *Perfective* maintenance, and adds *Emergency* maintenance as a fourth category.

Definitions according to **IEEE[3)]**:

- **"*Corrective maintenance*** is a reactive modification of a software product performedafter delivery to correct discovered faults.
- ***Adaptive maintenance*** is a modification of a software product performed after deliveryto keep a computer program usable in a changed or changing environment.
- ***Perfective maintenance*** is a modification of a software product performed after deliveryto improve performance or maintainability.
- ***Emergency maintenance*** is an unscheduled corrective maintenance performed to keep a system operational."



Ref: Schach 2002 p.12

The reasons why software maintenance is needed

Many motives are initiating various alternations to be done in a software system[4], among which the following reasons are the most common:

- functioning of the software must be maintained (perfective maintenance);
- faults in technical characteristics, design and realizationhave to be corrected (correctivemaintenance);
- ultimate product has to operate in various environments (adaptive maintenance);
- ultimate product has to evolve (evolution maintenance).

II. SOFTWARE MAINTENANCE- COSTS JUSTIFICATION

As said, most people being non-IT professionals, usually have blur knowledge of the term "software", and consequentially have wrong belief that maintenance of the software has to do only with the correction of errors. This fame is contrary to the results of different studies, which clearly prove that more than 80%of the maintenance effortsare used for non-corrective actions, i.e. approximately 75% of the maintenance effortswere spent on Adaptive and Perfective types of maintenance activities. Furthermore, the studies show that the error-correction maintenance activities constitutearound 21% of the overall maintenance efforts.Hence, the numerous software maintenance study reportssuggest that the cost of maintenance can be up to 67% of the cost of overall Software Development Life Cycle (SDLC). In average, the software maintenance costs cumulate more than 50% of all SDLC phases.

Before going any further in explaining the software maintenance cost's determining reasons and factors, let'sgenerally consider: when it can be said that an "expense is reasonable"; which are the similarities and differences between software development and software maintenance; what is the purpose of software maintenance; what software maintenance actually is comprised from, respectively which activities compose them.

Generally speaking, an expense is being considered as "reasonable" if:

- Resulting benefits are bigger than expenses;
- expense stipulates continuation of the normal system functioning, so consequentially contributes to its further improvement;
- when the expenses are initiated by legal demands.

1. Differences between Software Maintenance and Software Development (Software Maintenance versus Software Development)

It will be beneficiary for those without any software maintenance work experience, to clarify the difference between management of maintenance activities, and management of software project activities.While[5]project management is arranged with regard to the delivery of a product inside a particulartime period, and a scheduled project closure time, on the other side the maintenance phase must be organized to face daily work for its clients with a uninterrupted reliable service and by default, no end-date. In that respect, the Software Development is a sort of a newly product for which a brand new environment and system is produced from the scratch, usually within sketchy boundaries, of course if not otherwise limited by purchaser's request.In contrast, the Software Maintenance term applies to all the modifications and all the updates made after the commence in live of the software product, and as such and presents important phase of SDLC.

Software Maintenance results with:

- Improved performance;
- Corrected faults;
- Adaptations to a modified environment; which all together actually represents the core intention of maintaining.

Commonly both, software development and software maintenance, implicate the making of working code by means of the efforts of human developers who have appropriate capability, instruments, and methods. Conclusively, key difference between software maintenance and software development is consideration of restraintsinflicted by the existing system and by the existing environment. From the historic prospective, the Software maintenance hasn't received the same level of consideration as have received the otherphases. This is mainly because the Software maintenance is being the most mistreated field of software engineering. Hence**,** in most organizations around the World, software development has had a much higher profile than maintenance. Intention of software development is by producing code to resolvesetbacks or to gain business benefit. By comparison, maintainersreopen developed products, while also investigate the present by working with users and operators.Throughout maintenance, it is necessary systematically and completely to investigate and understand someone else'spast work and afterwardsto accomplish the demandedchanges and modalities. In addition, maintainers are also focused ahead anticipatingpredictable problems due to envision functional modifications. Practically, the software maintenance may extend over 20 years or so, whereas software development usually averages 1 to 2 years.

2. Software maintenance activities

Prof. Meir M. Lehman in 1969 was the first to address the Software maintenance[6] and evolution of systems. Since then, his research led towardsarticulation of the so-called Lehman's eight Laws of Evolution[7]. Lehman noted "really, the maintenance is an evolutionary development and that maintenance decisions are ministered by comprehending what happens to systems (and software) throughout time, that really maintenance is a constant

development, except that there is an extra input, respectively constraint – the existing software system, and that large systems are never complete and endure to evolve. As they evolve, they grow more complex unless some action is taken to reduce the complexity".

In a software context and in daily usage, the term "maintenance" can apply to some 21 types of changes to existing software, but the 2 mostly widespread understanding of the termapplies to:

- Fixing faults;
- Improving existing or adding new functionalities.

Due[8)] to complicity and inclusion of so many various types of activities under the term of "maintenance" companies usually justgroup together all forms of maintenance in order to use lump metrics such as the gross percentage of annual software fundsdesignated to all types of maintenance summed all together.

Table II Types of activities staged under the "Software Maintenance" word

1. Major Enhancements (new features of > 20 function points)
2. Minor Enhancements (new features of < 5 function points)
3. Maintenance (repairing defects for good will)
4. Warranty repairs (repairing defects under formal contract)
5. Customer support (responding to client phone cells or problem reports)
6. Error-prone module removal (eliminating very troublesome code segments)
7. Mandatory changes (required or statutory changes)
8. Complexity analysis (quantifying control flow using complexity metrics)
9. Code restructuring (reducing cyclomatic and essential complexity)
10. Optimization (increasing performance or throughput)
11. Migration (moving software from one platform to another)
12. Conversion (Changing the interface or file structure)
13. Reverse engineering (extracting latent design information from code)
14. Re-engineering (transforming legacy application to client-server form)
15. Dead code removal (removing segments no longer utilized)
16. Dormant application elimination (archiving unused software)
17. Nationalization (modifying software for international use)
18. Year 2000 Repairs (date format expansion or masking)
19. Euro-currency conversion (adding the new unified currency to financial applications)
20. Retirement (withdrawing an application from active service)
21. Field service (sending maintenance members to client locations)

Most commonly, changing an existing application is preferable option, rather than beginningdeveloping new application from scratch.

3. What is Software Productivity?

By default[9)], from economic prospective the ratio between the produced quantity of goods or services versus inputted labor or costs for producing them is called the "productivity". According to the aforementioned definition, from Software engineering prospective, the "software productivity" is understand as the ratio between the produced quantity of software versus inputted labor or costs for its production. After definition of software productivity, the next imposed question is what to measure. Regardless for what type of software project (development project or maintenance project) we are talking about, and regardless what type of software metrics is used, we always measure the effort respectively the productivity e.g. we convert the software size into effort in man-hours or man days using a conversion factor (known as the productivity figure), further we convert this effort into money also using a conversion factor.Goal of this paper is not to discuss about the software metrics but instead to help in the proper understanding of software project expenses.

Contrary to easily countable lines of code and pages of documentation, the software functionalities do not have anystandardized unit of measure and therefore are harder to quantify.Hence, required is establishment of a Software metrics usablefor quantifiable measure of various specifics of a software system, software development, and software maintenance process. Such metrics should be capturing the required producing effort for development or maintenance of a software, as well capturing of the provided software functionality.When measuring software productivity considered should be application performance too, which means how fast the software resolves any particular difficulty.Numerous factors affect the speed of an application. Some affecting factors could be managed by developer, while some others could be linked to hardware and operational environment dependences. Memory usage, code portability and reliability[10)] can be yet another performance measurements. 1940s are recognized as the beginning of the computer era, and ever since Software project cost estimation is an important but difficult task.

Generally speaking, calculating the Software cost means calculative value of effort respectively the productivity. Time spent for development or maintenance of a software application is in close relation with its size and how complex is the solution itself. Also many additional influences affect estimation of software effort like technical environment, developers' expertise, maintainer's experience, and availability of development instruments. Along with increment of such influencing factors, increases also the degree of needed effort for development and maintenance of software.

Consequentially such influences which increase maintaining efforts, shall inevitably increase project price, due to direct correlation between maintenance cost and IT professionals' labor costs.

Further below is list of some influences which make discernment between new software development versus software maintenance, resulting with increased maintenance costs[11)].

Team stability: Companies usually do not have clear differentiation between software engineers when considering their roles of either "new project developers" or "maintenance engineers". Because software maintenance may extent over 20 years or more, while development may extent over 1 to 2 years, therefore it is hard, difficult and demanding for the maintainers when they inherits the product (in absence of originated developers)to understand the system and its code. Maintainers have to take the developed final product, its code and documentation, which are often missing or incomplete because of the mentioned reasons.

Contractual responsibility: Although usually vendors are pressing clients for a maintenance contractstogether with the software development ones, never the less in most cases the maintenance Agreement is apart from development Agreement. Because of this, and in conjunction with the low level of team stability, there is no interest for developers to write easily maintainablesoftware. It is expectable that since developers are not responsible for maintenance, they tend not to care a lot for that. Even more, because of this "tendency", developers are highly personally motivated to save their effort during development process.

Staff skills: In most of the cases, even when highly skilled debuggers and coders are required for an effective maintainers, often it is allocated to the novice junior staff.Even more, legacy systems usually are written in obsolete programming languages, so therefore the maintainers may not have abundant, if any, knowledgeof these languages.

Program age and structure: Bulk of the software now in use is more than 10, and some applications are even over 25 years old. Result of that situation is that year by yearthe maintenance of such "elderly" software, inclines becoming increasingly problematic, since updates progressivelyobliterate the initialassembly of the applications by creating layers of new technologies around the original essential software system. In the absenteeism of today'sstate-of – the-art software engineering methods, many old systems have been developed without anyhelpful automated tools and techniques. Most of legacy systems were never well structured and were usually optimized for efficiency rather than clarity and understandability, as was the emphasis years ago when RAM memory and speed were much more significant matters. Difficulties[12)] most usually related with old systems are inadequate documentation, unstructured (so called "spaghetti code" with obnoxious control structure), and lack of personnel knowledgeable about the product.

Stressful sort of work: Two fastest changing aspects of our world are the Technology and the Business environment. As most of maintenancearebeing executed in live production environment, so responding to the business requirements in a timely manner becomes stressful requiringwith high level of responsibility and accuracy. Regular day-to-day maintenance is important to insure that:

- business is always taking optimal advantage of what the used software is capable of;
- used software matches the business' current demands; and
- used software ensures business efficiency by keeping up the business with market trends.

Fluctuation in the workload and resourcing problems: Maintenance is usually consisted of two significant type of work:a) improving the application software by appending new functionality, and b)solving bugs or other malfunctioning issues raised by the client.Neither a) nor b) are perpetual and continual even the first seems to be so. Due to the aforementioned reasons is very difficult to predict and planning the exact workload even week in advance. From resourcing point of view, the team size cannotbe changeddepending on workload regardless of how stressful it is, the existing team will have to take the fluctuating workload.

Why software maintenance is so expensive: Here are some other reasons why maintaining software is so expensive. Not only that human labor cost a lot, but the tools they tend to use are also expensive too.Maintainersevermore seem to require more time than was originally envisioned at the beginning.

Software by its nature is constantly evolving: Because software is a model of reality and reality is permanently changing, except thatclientsrequest for new and modified functionalities, also the systems environmentwhere the software operatesis constantly modifying, causing the software itself to be modified due to fit the newlychanged environments. Buying software does not automatically means the full recognition of his functions by the users. For this reason, it is necessary to prepare staff for using of its full functionality, so that the benefits from the utilization of the software to be maximum.It is inconceivable nowadays to imagine any enterprise without adequate software.So in terms of costs and benefits for enterprises is a good idea all activities about software maintenance (making updates and upgrades) to perform the vendors who needs to keep a stable team of maintainers monitoring how customers are using the software and what type of support they require.In case of failure,users usually do not know did a software bug cause it, ora system breakdown, or something erratic in the network infrastructure. Therefore inevitable are IT experts for diagnosing failures and pursuing the solutions regardless from failure's nature. Hence, in front of managers of enterprises there are two possibilities: either to pay vendors for proper maintenance of their enterprise software, or to employ IT specialists who will proper maintain the software system of the enterprise. There is no doubt that the right answer to find the proper choice which of both potential solutions is better, comes after one cost-benefit analysis is made.

Another emerging option for avoiding all of these expenses is by using software in the cloud. It is a possible to decrease costs by renting software (SaS) which runs in the "cloud", with client's:

- agreeing to use the standardized version of the software that everyone else is using too, and additionally to pay extra for the tweaking, which as result will increase the operationalcosts; and

- accepting and bearingwith the changes that come when the cloud-based software vendor chooses to upgrade the system – on the vendor's schedule, not users'; and
- awareness that the cost savings of the cloud are not completely one-sided.

Hence, it might happen that those IT specialists for maintaining are smarter choice after all.

Some reasons and factors for software systems becoming hard to maintain over prolong period of time are:

- **Costs of understanding or program comprehension** asit is of essential importance that the maintainer obtains acomplete understanding of the structure, behavior and functionality of the software applicationeach time a modification is done into software's source code.Understanding[13] means how quickly a software maintainer can understand where to make a modification or fixes in software.Consequentially, maintainers spend a largeamount of their time reading the source code and its accompanying documentation, in order to comprehend its logic, purpose, and structure. Some analyses show that 40% to 60% of the maintenance effort is dedicated to this task sole.One way to overcome the incomplete and imperfect understanding is to promptly produce documentation and other supporting specifications;
- **Lacking or incomplete documentation**, as well as lacking people knowledgeable of the software due to their leave or retire without being replaced.It is well-known software systems' tendency to grow and expand down the road, becoming more complex. Suchever-evolution makes the systems more difficult and by that even more costly to maintain.Overall re-Engineering of a system usually is not quite an option due to: a) the scale and volume of the system that needs to be replaced, b) the test coverage unknown, and c) the original and modified requirements are not well documented;
- **Impact analysis**, which is the action[14] for evaluating the possible results of a modification targeting to minimize unexpected and undesirable secondary effects. It also includes the detection of the system's portions that need to be changeddue tosuggested modification.

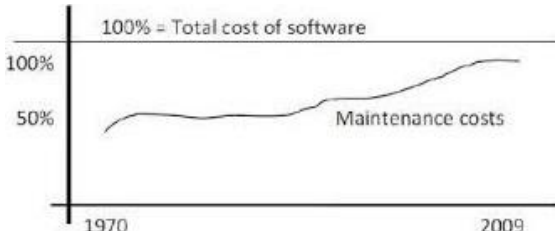


Figure1. Expansion of Software maintenance costs as percentage of overall cost

After implementation of modifications, the software system must be tested to insure that it will perform according to the specification. Testing[15] a system after it has been modified is called "regression testing". Goal of such regression testing is to verify that modifications are correct and to confirm that unchanged pieces of the system have not been influenced;

Other involved cost factors in software maintenance, besides those mentioned above, are:

- deficiency of programming standards and guidelines;
- resolving bugs and glitches;
- implementing modifications;
- interaction and coordination;
- softwareoldness and structure; and
- communication and data exchange with other systems.

Expenses for maintain aging software typically contribute 75% into TCO[16]. Split by categories they include:

- **Corrective maintenance** – Expenses caused by software changes due to correct problems discovered after initial deployment (usually 20% of software maintenance costs);
- **Adaptive maintenance** – Expenses caused by software changes due to keep it effective in a ever-changing business environment (usually 25% of software maintenance costs);
- **Perfective maintenance** – Expenses caused by software changes due to advancing or enriching a software solution in order to improve its overall performance (usually 5% of software maintenance costs);
- **Enhancements** – Expenses caused by software changes due to continuing innovations (usually 50% or more of software maintenance costs).

At the end of this list of factors and reasons which directly influence the price of "high" maintenance of software we are presenting the list of activities which are the responsibility of the maintenance team, and a list of some factors[17] which are affecting maintenance effort-productivity.

Table III Some factors which are affecting maintenance effort and productivity

Maintenance team responsibilities
Understanding the system
Location information in system documentation

Keeping system documentation up-to-date
Extending existing functions to accommodate new or changing requirements
Adding new functions to the system
Finding the source of system failures or problems locating and correcting faults
Answering questions about the way the system works
Restructuring design and code components
Rewriting design and code components
Deleting design and code components that are no longer useful
Managing changes to the system as they are made

Factors affecting maintenance effort-productivity

Application type
System novelty
Turnover and maintenance staff ability
System life span
Dependence on a changing environment
Hardware characteristics
Design quality
Code quality
Documentation quality
Testing quality

Usually non-software practitioners being asked about software maintenance would instantly think about removing or fixing software glitches, as making changes in an operatingsoftware occurs since the beginning of computer era. The Lehman's first law[18),19)] of evolution points out that if a software systems want to be successful and competitive, than it must change over time: "A program that is used in a real world environment necessarily must change or become progressively less useful in that environment". Considerable modifications also arise from the necessity to adjust software to cooperate with ever changing external environment, i.e. people, companies, and other electro-mechanical systems.

Nowadays because any company is being fully dependent on software systems (which by definition with the time do not "wear out", but simply become less useful), so consequentially companies are imperatively investing big money in software's maintenance and evolution.Companies' software systemsare critical business assets and because of that, companiesare forced to spend on system modifications in order to preserve the value of the software resources. As ever since 1972[20)],maintenance of software applications was little comprehended, so usually it was referred as to an "Iceberg" aiming to highlight the hugeaccumulation of potential difficulties and expenses that lie under the surface. Due to large amounts of finances are poured into maintenance, therefore any new definition of software maintenance has to be beneficial, and because within such bigexpenditure, even a small technical advances must be worth many times their cost.

However, mitigating software expenses within the common business people is itself still a little bit inferredtasks. Conventionallyis to evaluate investments towards their expense and envisioned income, but the expected income of a program which supports a business without directly generating revenue is difficult to calculate and is currently the subject of research[21),22)].Cost of software maintenance for a company can be substantial, but yet maintenance processes are not assessed by the same investment analysis as decisions in other areas (including software development).

Normally maintenances are managed trough fixed budgets where calculation of returns on investmentsis not performed at all.Software maintenance to be considered not as a costly chore but as an investment activityinstead, a cost/benefit analysis has to be performed, where the net benefits of maintenance would be measured (or envisioned) as the effects on the company's profits as a result of the changes and expressed in monetary units, as an comparative advantage or more generally as a return on investment.Treat from such prospective, it can be freely concluded that maintenance is an investment activity, rather than simply a costly chore.

Finally as support of what was said above, we will briefly mention some of the numerous benefits arising from the signing of SLA (Service Level Agreement) between vendors and users of their services. Vendors often try to push for maintenance contracts along with software development. Such Maintenance Contract would usually be calculatedand chargeyearly established on some percentage of the total software (development) cost.

Generally it charges for overall annual support and maintenance of a software product, including telephone assistance time and user support.Users might be unsure whether to go ahead and sign up for them for the reason that there are opinions that say Vendors misuse their positions by benefiting more from higher maintenance fees. One of the most significant benefits arising from such a contract for the users from thefinancial point of view is the ability to get control over their software expenses. If software bugs and regular upgrades are covered for then the overall IT costs should be reduced to a single monthly (or annual) maintenance fee. This eradicates the predicting game of IT costs and removes big unforeseen upfront expenses over a prolong period of time and will enable the company, to concentrate on its core business activities. In each industry, maintenance personnel is greater than those building new products.

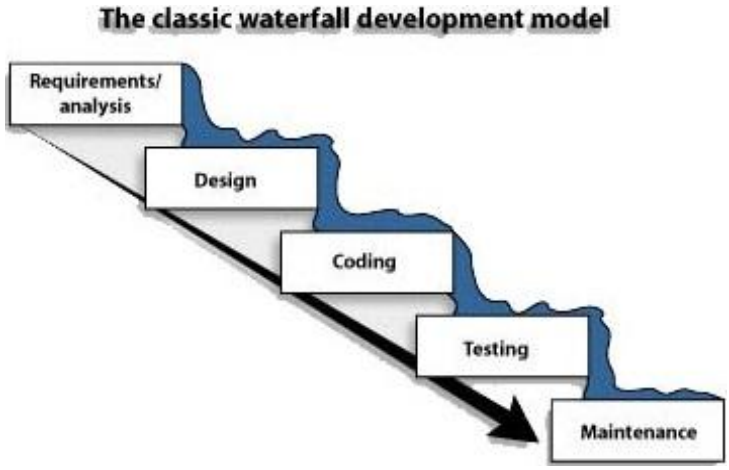


Fig.2 The classical Software Waterfall development mode,

In addition, the software industry[23)] which is labor very intensive, is not exception from this fact, because the number of maintenance personnel is unusually large and may achieve the number of more than 70% of all technical software workers.Except that Table.4 shows a drastic increase in the number of personnel engaged in maintenance of software over the past years and the expected trend in the following years, it also represents underestimating that has been made to the software-maintaining phase throughout these past years**.**

Table IV Software personnel in U.S. assigned to Development and Maintenance

Year	Development Personnel	Maintenance Personnel	Total Personnel	Maintenance Percent
1950	1.000	100	1.100	9.09%
1955	2.500	250	2.750	9.09%
1960	20.000	2.000	22.000	9.09%
1965	50.000	10.000	60.000	16.67%
1970	125.000	25.000	150.000	16.67%
1975	350.000	75.000	425.000	17.65%
1980	600.000	300.000	900.000	33.33%
1985	750.000	500.000	1.250.000	40.00%
1990	900.000	800.000	1.700.000	47.06%
1995	1.000.000	1.100.000	2.100.000	52.38%
2000	750.000	2.000.000	2.750.000	72.73%
2005	800.000	2.500.000	3.275.000	76.34%
2010	1.000.000	3.000.000	3.800.000	78.95%
2015	1.250.000	3.500.000	4.500.000	77.78%
2020	1.100.000	3.750.000	4.850.000	77.32%
2025	1.250.000	4.250.000	5.500.000	77.27%

III. CONCLUSION

In nowadays of ever changing demand, which comes from dramatically, evolution of business and technological world, software maintenance is the process of prompt adjust and enhance value of a software application through its modifications.Historically,software maintenance has unfairly suffered from negative perception towards it. It is possible to change these negative attitudesby providing the customer with in-deep view into maintain process and activities performed by the maintainershopping towards a clear mutual agreement on the results of these activities.

Drastic increase in the number of personnel engaged in maintenance process,is direct result of the enormous growth of the production of software, and as consequencethere is rise of the number of research publications related to it.All these together in the recent years have changed the negative attitudes about Software Maintenance. Between non-practitioners of software maintenance, the reasons for negative attitudes in relation to the high level of expenditure for the software maintenance are located partly in the sense of not complete understanding of the term Software, and mostly in not being familiar with the process and complexity of activities that include Software maintenance phase.

Software cost estimation is the estimation of effort, respectively the productivity of developers and maintainers.Software maintenance is costly because it has a number of various and complex activities which require people to work for years, and software professionals are expensive. Generally, software estimating is a difficult task. Especially estimation of maintenance is even more difficult and more complex due to complex relationship between the application that is being modified and the changes being made to implement new features or repair bugs.

In addition to consumption of a large part of the overall lifecycle costs, software maintenance is important because it aids flexibility of the business, as one of the main factors for its survival.

Not changing software quickly and reliably results in losing a business opportunities and loss of competitive ability of the enterprise.

Since years ago, software maintenance was usually described as an “iceberg” to draw attention to the huge volume of potential difficulties and expenses that lie beneath the surface.

In order software maintenance in organizations to be considered as an investment activity, there is a need to perform a cost/benefit analysis, where the net benefits of maintenance operations would be treatedas the effects on the organization's profits and presented in monetary units, as an comparative advantage or more generally as a return on investment.

Some of the numerous benefits arising from establishing SLA (Service Level Agreement) between vendors and users, are that this SLA usually are calculated as an annual fee based on some percentage of the total software cost, generally providing for overall support and maintenance of a software products including telephone assistance time as well.If users are covered for software glitches and regularly receive upgrades, their overall IT expenses will be reduced to a single fee – maintenance fee.

REFERENCES

[1] Lientz, B. P., Swanson, B. E., "Software Maintenance Management", Addison Wesley, Reading, MA, 1980.
[2] Pigoski, T. M., "Practical Software Maintenance – Best Practices for Managing Your Software Investment", John Wiley & Sons, New York, NY, 1997.
[3] IEEE Std. 1219-1998, "Standard for Software Maintenance", IEEE Computer Society Press, Los Alamitos, CA, 1998
[4] http://www.sce.carleton.ca/faculty/ajila/What-if.pdf
[5] http://www.researchgate.net/publication/228433114_Software_Maintenance_in_a_Service_Level_Agreement_Controlling_the_Customers_Expectations
[6] http://sce.uhcl.edu/helm/swebok_ieee/data/swebok_chapter_06.pdf
[7] M.M. Lehman. Laws of Software Evolution Revisited, EWSPT 96, October 1996, LNCS 1149, Springer Verlag, 1997
[8] http://www.compaid.com/caiinternet/ezine/capersjones-maintenance.pdf (12-2015)
[9] Jones, Capers. Applied Software Measurement: Assuring Productivity and Quality. 2 ed. McGraw-Hill, 1996.).
[10] Jalics, Paul J., and Santosh K. Misra. Measuring Program Performance. Systems Development Handbook (P. Tinnirello, ed). 4 ed. CRC Press LLC, 2000
[11] http://jroller.com/njain/entry/why_software_maintenance_is_so (09-2015)
[12] http://www.nptel.ac.in/courses/106105087/pdf/m14L36.pdf (09-2015)
[13] http://asq.org/public/wqm/how-to-save-on-software-maintenance-costs.pdf
[14] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.25.1678&rep=rep1&type=pdf
[15] Leung, H. K. N., White, L. J., "Insights into Regression Testing", Proceedings of the Conference on Software Maintenance, Miami, Florida, IEEE Computer Society Press, 1990, pp. 60-69
[16] http://galorath.com/software_maintenance_cost (12-2015))
[17] http://www.cs.umd.edu/~mvz/cmsc435-s09/pdf/slides14.pdf *(09-2015)*
[18] Lehman, M. M., "Lifecycles and the Laws of Software Evolution", Proceedings of the IEEE, Special Issue on Software Engineering, 19:1060-1076, 1980.
[19] Lehman, M. M., "Program Evolution", Journal of Information Processing Management, 19(1):19-36, 1984. [Costs. Journal of Computer and Communications, 2, 1-16.
[20] Canning, R., "The Maintenance Iceberg", EDPAnalyzer,10(10), 1972
[21] E. Clemons. Evaluation of strategic investments in information technology. CA CM, 34(422-36, January 1991.
[22] R. E. Whiting, J. Davies, and M. Knul. Investment appraisal for IT systems. British Telecom Technical Journal, 11(2):193-211, April 1993
[23] http://www.compaid.com/caiinternet/ezine/capersjones-maintenance.pdf (*12-2015)*

FIGURE REFERENCES

Fig1. Development of Software maintenance costs as percentage of total cost Floris P, Vogt Harald H. How to save on software maintenance costs, Omnext white paper, SOURCE 2 VALUE, 2010.
Fig2. http://rootsitservices.com/CustomPages/sdlifecycle.aspx

TABLE REFERENCES

TABLE I. http://www.worldscientific.com/doi/suppl/10.1142/5318/suppl_file/5318_chap1.pdf
TABLE II. http://www.compaid.com/caiinternet/ezine/capersjones-maintenance.pdf
TABLE III. http://www.cs.umd.edu/~mvz/cmsc435-s09/pdf/slides14.pdf
Table IV. http://www.compaid.com/caiinternet/ezine/capersjones-maintenance.pdf

