Retirement of Software: Another Milestone in SDLC

Deepak Jain  
HCL Technologies Ltd.  
A-5, Sector-25, Noida (UP), India  
E-Mail: j.deepak@hcl.in

Shalini Jain  
Lecturer  
Maharaja Surajmal Institute of Technology,  
New Delhi, E-Mail: shallu.jain@gmail.com

ABSTRACT
The software marketplace is in a constant state of change with new and disruptive markets emerging while some traditional markets fall by the side. The software development market is also characterized by an irrevocable trend towards greater software complexity -- complexities that not only complicate the community of software developers but also create financial pressures on them as well as their customers. Software technology is changing at such a high rate that software bought this morning gets outdated by next evening. In such a scenario, many a times consumer of the product suffers for paying for an older software unknowingly. It suffers with the system when developer organization of such old systems stops providing technical supports on them stating that newer versions are available in the market to solve their problems and hence the customer should discard the system and purchase a new one. In this paper, the author suggests some principles for formal announcement of retirement or exhaust of any software to make the consumer aware if company stops working on any of its product or project. The paper highlights the ethical responsibilities of the companies regarding software retirement and suggests some formal approaches to accomplish the task of retirement. The paper supports to take software retirement as a milestone in the development process and redefines some software development process models for incorporating software retirement in them.

KEYWORDS

INTRODUCTION
Software retirement is a post development activity in the software life cycle that is to be executed in a formal manner and with sincere efforts as any carelessness in the procedure may cause big harms to the consumers of the software. Dynamicity of software and hardware technology demands quick changes and enhancements in the systems. Amendment of the system is limited by its enhanciability which is the extent to which the system may be enhanced with affordable efforts and controllable risks. Finally, the system reaches its state of saturation where any more changes may result in complete crash of it or the alterations cause unaffordable costs and resource consumptions. At such states, the companies are expected to declare the product “dead”, and stop further working on it.

If declaration of software exhaust is kept up to the knowledge of the developing organization and the customer is not made aware of it, the customer may suffer with such exhausted products. Further, new deployments of these systems may be continued in the market through indirect dealers of the product and new stakeholders of the system may suffer paying for the system for which the company has already stopped working and stopped providing supports and updates.

In these concerns, this paper highlights the ethical responsibilities of software developing organizations about formal announcement of exhaust or decommissioning of any project or product. The author summarizes the expectations of the consumer market for a formal announcement for retiring a software project and suggests some formal approaches for execution of the process of retirement of the project.

MANDATORY STATE OF EXHAUST FOR ANY SOFTWARE
Software life cycle normally follows the steps from requirement engineering through designing, coding and testing and usually ends into its maintenance phase. The enhanciability of software limits the permissions to modify it and in turn limits its life. In a finite time, the software reaches to an extent where the cost of enhancement shoots unaffordable and the failure rate rises uncontrollable. Such a state of the software may be understood death of software and named as “Software Exhaust” (at least for the context of this paper, if not globally). Software exhaust indicates the time of retiring the software and suspension from usage or new deployments. Limited enhanciability of the software ensures certainty of its death. In this manner, software life cycle should begin with an idea or need that can be satisfied fully or partly by any software product and it should end with the retirement of the software product.

SOFTWARE RETIREMENT PROCESS
As understood in the previous discussion, retirement of the software in a finite period is certain. The focus of the Retirement Phase is successful removal of a system from production when either it is no longer needed or reinstating it by another advanced system. Also the system may be required to replace to analyze its effect on other related systems or its coupling in the environment. Another reason of it may be
reconstruction or alteration in a super-system the part of which is this system. In retirement of the software, the software product is removed from production, often called "decommissioning". It can either be replaced or simply no longer supported. The release of a new version of software often coincides with the retirement phase of an older version. Scott W, Ambler, in its Enterprise Unified Process (EUP), has mentioned the following major activities in a formal retirement phase:

a. **Analyzing system interactions:** Being the system a complex combination of small and interacting software, the part that is to be decommissioned is required to be analyzed thoroughly for its interaction in its environment. It is analyzed as a black box for structure of its inputs and outputs through which it is coupled with its surroundings. Dependencies of the coupled systems on the retired system are scrutinized thoroughly and effects of decommissioning of the software are explained unambiguously.

b. **Determining retirement strategy:** There may not be a general rule for retiring of the software irrespective of its nature. Retirement strategies may differ with nature and scale of impacts. Any new release of the system may need simple data conversions and database replacement operations. For some in-house construction, the complete designs are to architecture and the developed system is to be implemented thoroughly. A new version of the software may need creation of simple files, copying of data, nominal modifications in structures and user interfaces. Complete retirement of a system may be the hardest case because huge rework on the external systems is required and this may even lead to retirement of other systems that are not compatible with the newer system.

c. **Updating the documentation:** Simple replacement of old software with new does not closes the retirement process. A wide range of documentation, including operations and support procedures, enterprise architecture models, system portfolio documentation, administration documentation, and system overview documentation etc are also required to be updated. In future, the updated documents are required to be referred to work with the new system.

d. **Testing Thoroughly:** The migration tools must be tested in the same way as to test a system during the transition phase of its development cycle. The replaced part must be tested thoroughly with its couplings in place. Compatibility and correct behavior of the surrounding components of the new system must be taken on records.

e. **Migrating users:** The migration or retirement of the system cannot be accomplished abruptly. Major risk in success of new system lies in its acceptability with the user. The end user is required to be notified appropriately about this. The user may also be provided with technical and non-technical assistance for migrating from the old system to the new one. This may also need conduct some special trainings for the user.

f. **Archiving:** The existing data, code, documentation, and other system artifacts must be properly archived so that it may be restored at a future date if required.

g. **System removal:** This is often a complex task, as all the important data must be migrated or converted, access is turned off and then removal of all vestiges of the retiring system is accomplished. System may be removed in a single or multiple phases depending of the scale of retirement.

h. **Updating environment and other systems:** As discussed earlier, the system does not exist in isolation and for this, while retiring a system, its environment is required to be modified and upgraded. All the other systems interacting with the retiring system need upgrading. Some supporting systems may also require be removing or replacing with newer systems. Finally the complete environment is updated in accordance with the changes incorporated in the system.

**CURRENT SCENARIO OF SOFTWARE-EXHAUST ANNOUNCEMENT**

Software development companies develop, test, install and provide training for the user to work on the software product. During its use, the developing organization is responsible for debugging the software for the new discovered errors and bugs in the software those were left undiscovered during its testing. In order to enliven their relations with customer, the developing company may even update the software for a limited course of time, accommodating the changes in user requirements and technical up-gradations in its environment.

The companies are found to show fair interest in the software and trying to provide their best in order to meet the consumer expectations and their satisfaction for a course of time. But unfortunately it does not last long. Slowly and gradually, the company looses its interest to enhance the system as it accounts
for big hassles and long insecure efforts. Working on aged software involves higher risks of failure. This should be taken ethical on company side to declare their software dead when it reaches to the state of “Exhaust”. Ethically, any developing company is expected to announce a formal retirement of its product when it reaches to its state of exhaust. Declaration about decommissioning of software, if made from customer or user side is one thing that may not be that objectionable because in this case, the stakeholder of the product or the customer is not in any hidden or unannounced harms. But if decision about retirement of a software project comes from the developing organization, i.e. when the developing company decides to stop production and support of any of its product, then the consumer may suffer undeclared immediate or long-term losses because of unplanned or unannounced informal retirement of the product. Due to such informal retirement of the software, the consumers suffer the loss of information and money.

Software companies generally do not give any formal declaration for exhaust of any of its software products although it happens for all software. In some (one in thousands) cases, even if some companies announce the retirement of any of its product, they do not publish any formal report for this that would contain the information about the facts like major reasons of retirement, technical limitations the product is facing and the focus or planning of retirement of the product. Generally, companies silently stop producing and maintaining its so called retired product because updating it seems to be too costly to afford and the failure rate gets increased frenziedly. Companies and their dealers usually retire their products silently to preserve their state of market. This may be termed as a silent death of the software or even a breed of it. The problem becomes even worse in case of retail and original equipment manufacturer (OEM) and market software product where the product is used by mass community. In such a case when the information is not published formally, users across the world suffer. The similar problems occur with manufacturers of hardware computer parts too. Many times, companies stop manufacturing some particular hardware or any of its models. At the same time, they stop updating its software drivers too. A problem of unavailability of updated drivers for hardware modules to use them with newer operating systems is very common in user community. Operating systems change at a very fast rate e.g. WIN9X series products were quickly replaced by WIN-XP and WIN-2000. But hardware does not vanish so fast and they need to be updated for their drivers. The problem persists with hardware that have been stopped in production and the companies show no interest towards the customers who paid them just a short time back, specially in case of hardware like internal modems and LAN-cards.

**EXPECTATIONS OF CONSUMER/MARKET**

It should be ethically noted that the customer (and also consumer in many cases) is charged a reasonably high amount as the cost of the software and license for their commercial and non-commercial usage. They also pay the costs of services of training and supports. During the process of snapping the tender of project from its competitors, the customer is assured for the best and uninterrupted timely services by the developing companies. The customer is provided with descriptions of a number of past successes stories hiding all failures histories and their old products for which they have stopped working upon. One could never name any single company that ever mentions and addresses the problems of their customers of any of their old products that they had stopped working for. Also not many companies are working to help their customers get rid of these problems.

It may be expectable and acceptable and even bearable to replace such software or drivers (or even complete hardware & driver set in some cases of acceptable cost) with another suitable. Needless to say that, the companies do not follow such strategies of replacing their old products except in some special cases of legitimate values. Moreover, when these companies execute exercises of replacing, they do not follow any formal and ethical manner. Consumer is charged for the new developments and only an inferior or no documentation at all is prepared about the reasons of replacing the software and analysis results of replacement. The consumer is never described about why the software is replaced and why the last one could not be enhanced as per its needs. Usually companies do not provide formal declarations about selected replacements too.

In future, it becomes hard to retrieve the reasons of replacement of the existing software with altogether new software and the process become tough to trace. Also, the consumer who has spent money on the development and maintenance of the previous software is expected to have a right to know why the company now has stopped providing updates to that software and what can be the cost of further upgradations. The problem becomes worse in case of OEM and market-products where the product is under usage of a bigger community across the glo. The companies are ethically required to publish the information about their decisions of no more working for enhancements of a product so that the user community may look for an alternative well in time. It is accepted that running any software beyond time boundaries and making it adaptable to all types of changes in its environment may not be feasible for a company and they need to take vindictive decision of stopping the production and maintenance of such products. But at the same time they need to put in mind the cost the consumer has to bear for that.

Further, the developing company should publish the formal information about exhaust or retirement of any of its product. A formal report of exhaust or retirement of the software needs to be published on the responsibility of the developing company. This report must clearly be describing key reasons for declaring the software retired and stop spending and working on it. In details, it should be enlightening the technical bottlenecks of the product in updating it to cope-up with the current technology expectations. Further, it should be suggesting some
excellent alternatives of the product that may be used for replacing the retired product.
Not only this, the organizations are expected to provide a formal plan for disaster management against the probable (however small) losses due to retiring of the software. They should produce a public report describing the expected areas of disaster for related fields. Generally companies prepare these reports, but they do not provide it for public or customer feedbacks, which are expected for the consumer.

APPROACHES FOR SOFTWARE RETIREMENT
Once understood that retirement of the software in a finite time is certain, one should plan well in time for formal retirement of the software rather following its silent death.
Companies generally stop supporting the system silently as the use of the system decreases in the market. In many other cases, companies declare the retirement at a short notice where the consumer may not be notified about it well in time to minimize the losses.
A better alternative for retiring the software is to preplan for its retirement. This may be a little tough because of unpredictable nature of technology dynamics however can be planned while development of the software. This may be planned better to retire the software in multiple iteration rather than retiring it in a single announcement. The companies can first replace or retire a part of the system and then move to the next one. Such an approach will allow the consumer to slowly adopt the newer systems and leave the older easily and smoothly. Even the developing company will have enough planning and flexibilities for changes in plans of retiring the software. Hence multiple iteration retirement may be a good alternative for large and complex systems.
Another suitable approach is to run the system to be retired and the replacing system in parallel. While retiring the systems that handle critical business functionalities, the task of decommissioning is executed by running the system to be retired and the system that is replacing it in parallel for a period of time to ensure that the functionality is working properly in the new system. Only when the stakeholders are satisfied about the functionalities of the new system, the old system is shut down. This provides the ability to verify that the new system is functioning appropriately by comparing it to the results of the old systems and also the option to quickly revert back to the old system if something goes wrong. But, such cases will require additional hardware, work-power and other resources and may also lead to redundancy in the information and storages. Such approaches should hence be adopted in critical cases only.
Another approach for retirement may be the use of incremental development of the system with deletion of functionalities and parallel replacement of them with newer and enhanced functionalities. In such approach, not only the system is to be developed incrementally but also updated in parallel. Software development companies can also work with different teams, one focused for incrementally developing the system and the other focused for incrementally upgrading the system retiring its functional parts one by one.

DEFINING RETIREMENT AS A MILESTONE IN SDLC
One can understand that death of the software in a finite time is certain and its formal retirement can be preplanned. This can define software retirement as another milestone in software development life cycle (SDLC). This paper redefines three process models for this purpose.
A. Extended Waterfall model
Extended Waterfall model is a sequential series of steps of software development process. Waterfall model follows a non iterative acyclic procedure in order to produce the desired software product. The model follows a big bang approach in which the complete software is delivered in a single version. Extended Waterfall process may be described with retirement as shown in Figure-3. In this model, the steps of software development include requirement analysis, designing, detailed designing, unit implementation, unit testing, integration of tested modules, system testing, installation, operation, maintenance and exhaust & retirement. All these activities are performed in a defined sequential order without skipping. None of the activity can be revised once has been closed and passes the results to the next step for use. The complete process runs as a waterfall that neither can skip any point in fall nor can repeat to any point once traversed.
Retirement in waterfall model lies in the end of the process. This step includes planning, analysis and execution of retirement of the software. This phase follows the general rules of waterfall model for other steps i.e. cannot be entered before completion of previous steps and cannot go back. This model preserves its superiority over traditional waterfall model in that it has retirement as a milestone in the lifecycle of the software. This forces formal considerations for retiring the software rather allowing its silent death. This forces formal planning and documentation for retirement of the software.
Major drawback in this model for retirement of the software is that this does not support any planning of retirement before the exhaust state arrives. Another drawback is that this model does not support phase wise retirement hence may not be a good choice for a huge project.
B. Quick Development & Retirement (QDR) Model

QDR model is an extension of traditional rapid application development (RAD) model. RAD model supports quick implementation of an application by integration on reusable components. QDR model supports the development in the same manner of development but that with retirement as another milestone. As the development now is an integration of reusable components, its exhaust is driven by extensibility of the individual components. Hence for improving enhanciability of the integrated system, it may be required to replace any of its exhausted part with a new but compatible component. QDR executes retirement of the system in two broad steps. Former step is called interim retirement and later is final retirement.

In post installation maintenance phase of QDR model, the system is analyzed for finding the component(s) demanding alteration for enhancement of the application. If the components are enhanciable, enhancements are performed. Else the system is declared retired temporarily and process proceeds to interim retirement phase. In this phase, the temporarily retired system is analyzed and updated by replacing some selected components of it. This means that QDR supports component level retirement of the system i.e. retirement of individual components of the system. This can increase the life of the system. The system after updating is redeployed for use and maintenance phase continues. This creates a loop between maintenance and interim retirement phases which continues till the software reaches a state where it cannot be enhanced by enhancing or replacing its components or the cost and risk in doing so becomes unaffordable. This state indicates final retirement of the system, in the final retirement phase, the system is completely discarded and replaced with an entirely new system.

QDR focuses planning for retirement of individual building blocks of the integrated system rather than the complete system in a single go and gives benefits of extended life of the software as well as phase wise retirement of the system. Every iteration in interim retirement is documented formally for future reference. Formal planning and documentation is also completed for final retirement of the system.

Incremental Enhancement Model with Retirement Support model is supposed to be best suiting for huge projects supporting phase wise development and phase wise retirement of older versions. This model is described pictorially in figure-5. Incremental enhancement model with retirement support facilitates formal retirement of the previous versions of the software parallel to the development of the new versions. The previous versions are retired with the installation of the new versions. Planning of these retirements takes input from the designs of the previous and new versions of the system.

This model provides benefits of phase wise replacement of older versions as well as can be modified to allow parallel running of new versions with old versions for analysis purposes. The model provides a budget friendly and schedule friendly approach for development and retirement reducing the risks of failure. One can plan for a part of the system to be developed first for a trial run and later replace it with newer versions with formal planning and execution of retirement of older versions.
C. Incremental Enhancement Model with Retirement Support

![Diagram of Incremental Enhancement Model with Retirement Support]

ETHICAL RESPONSIBILITIES OF COMPANIES
Companies declaring retirement of any software must raise ethical responsibilities on their own for the probable losses due to that retirement. Software developing organizations must not declare the retirement of any software until unless a viable path of migration from the software exists for its consumers. Organizations need to adopt software-retirement as a milestone in life cycle of the software product. The developing companies, while announcing the retirement of any of their software product, should provide (if possible) a formal or approximate schedule for retirement of the software well in time that allows the consumer to settle and prepare its environment, infrastructure and other arrangement to minimize the losses. The companies must effectively discourage new deployments of a system that is going to be retired in near future. They should provide appropriate guidance and technical support to their customers and the consumers to minimize their losses. Companies should suggest the suitable replacements for the retired products for their consumers. Companies should also prepare and produce disaster reports for public and customers.

CONCLUSION
Exhaust of software in a finite time is certain. Consumer expects formal retirement of development projects of companies. Formal retirement of software can be preplanned in the life cycle of the software. A software development process model can be followed for formal retirement of the software. Process models can also be followed to take advantage of phase wise retirement of huge systems. Process models can also provide benefits of extended software life by component level retirement. Software companies should ethically take the responsibility of formal announcement of retirement of any custom or open market software project. They should provide the information well in advance and discourage new deployments of the product that is going to be retired in near future. They should provide appropriate guidance and technical support to their customers and the consumers to minimize their losses. Companies should suggest the suitable replacements for the retired products for their consumers. Companies should also prepare and produce disaster reports for public and customers.

FUTURE SCOPE
The paper has covered to update some of the SDLC models with Software Retirement as a milestone with formal guidelines to achieve them achieving more customer ease and satisfaction. The future steps would include executing software projects with these models and measure the performance of each model for different types of needs. Also there is a need of defining ethical rules more formally to make the software retirement smoother for the customer as well as the developing organization leading them to a win-win situation.

REFERENCES

Continued on Page No.
Retirement of Software: Another Milestone in SDLC

Continued from Page No.