Volume 9, Issue 3, March – 2024 International Journal of Innovative Science and Research Technology
ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24MAR2132

Decision Making on a Software Upgrade or Decommission with Data Mining and Machine Learning Techniques in Information Technology Industry

Ravikanth Kowdeed
Submitted to the Research Committee at the Swiss School of Business and Management

Abstract:- The Organizations have been investing more in Technology and Infrastructure spends like software upgrades, software renewals, software replacements, platform migrations etc., apart from investment in Business, People, and Processes. In this context, it is not an easy task for stakeholders to decide whether to go for a software upgrade or to replace it with another software.

There is no unified approach or solution to consolidate data and relationships of Information Technology Assets, Software Upgrades, Software costs, Software defects, Software Performance Metrics, Security issues, IT system versions, service level objectives etc. Due to this, the decision making of software upgrades and software decommissioning is a tedious process and takes more time and effort.

There is a need to build a solution that can integrate and validate the information like software assets, software upgrade success and failure likelihoods, cost benefit analysis of Cloud Computing, software metrics for fault prediction, software maintainability prediction results, Digital Transformation readiness and other related factors.

There is an opportunity to apply Machine Learning techniques in defining and deriving the success likelihoods on the following data: Systems and data integration, software assets compatibility, operational service level agreement breaches, quality assurance metrics, security issues, number of open defects, number of defect fixes, number of priority incidents, mean time to resolve critical incidents, expected cost increase in software maintenance, potential cost reduction with the software or hardware replacement etc.

This Research Proposal outlines the above mentioned to build a recommendation system aka decision tree namely Software Upgrades or Decommissions Life Cycle.

I. INTRODUCTION

The main objective of this research is to gather information from the Software Engineering Life Cycle stages and apply Pareto law on the metrics at various stage which states - 80% of consequences come from 20% of causes - while establishing relationship between the stages by executing Machine learning models on this big data. This outlines the influence of Software asset attributes, platform compatibility, Software metrics, Software versions and dependencies, Software defects on the Software upgradation or software replacement need. All this data is fed into the recommendation system proposed that helps in decision making of upgrade or decommission of any IT system to cater to today's Digital Transformation needs.

The *Systems Development Life Cycle, Wikipedia,* is referenced for the stages defined. The information needed for this research across the stages is gathered from the public web sites, software release documentation, organization case studies and feedback surveys conducted in the communities of practice and communities of technology interest groups.

- Software Requirements (Business objective, System needs, Software features, Hardware specifications, Cloud vs non-Cloud infrastructure supported).
- Software Cost (Invest vs Operate Cost i.e., installation, renewals, upgradation and decommission costs)
- System Asset Metadata (Software features, version information, service level objective, end of life date).
- Software Issues (Compatibility issues with other software and hardware, Security Vulnerabilities, Quality defects, integration errors)
- Software Execution metrics (configurations, change management, performance metrics and maintenance activity insights)

II. RESEARCH METHODOLOGY

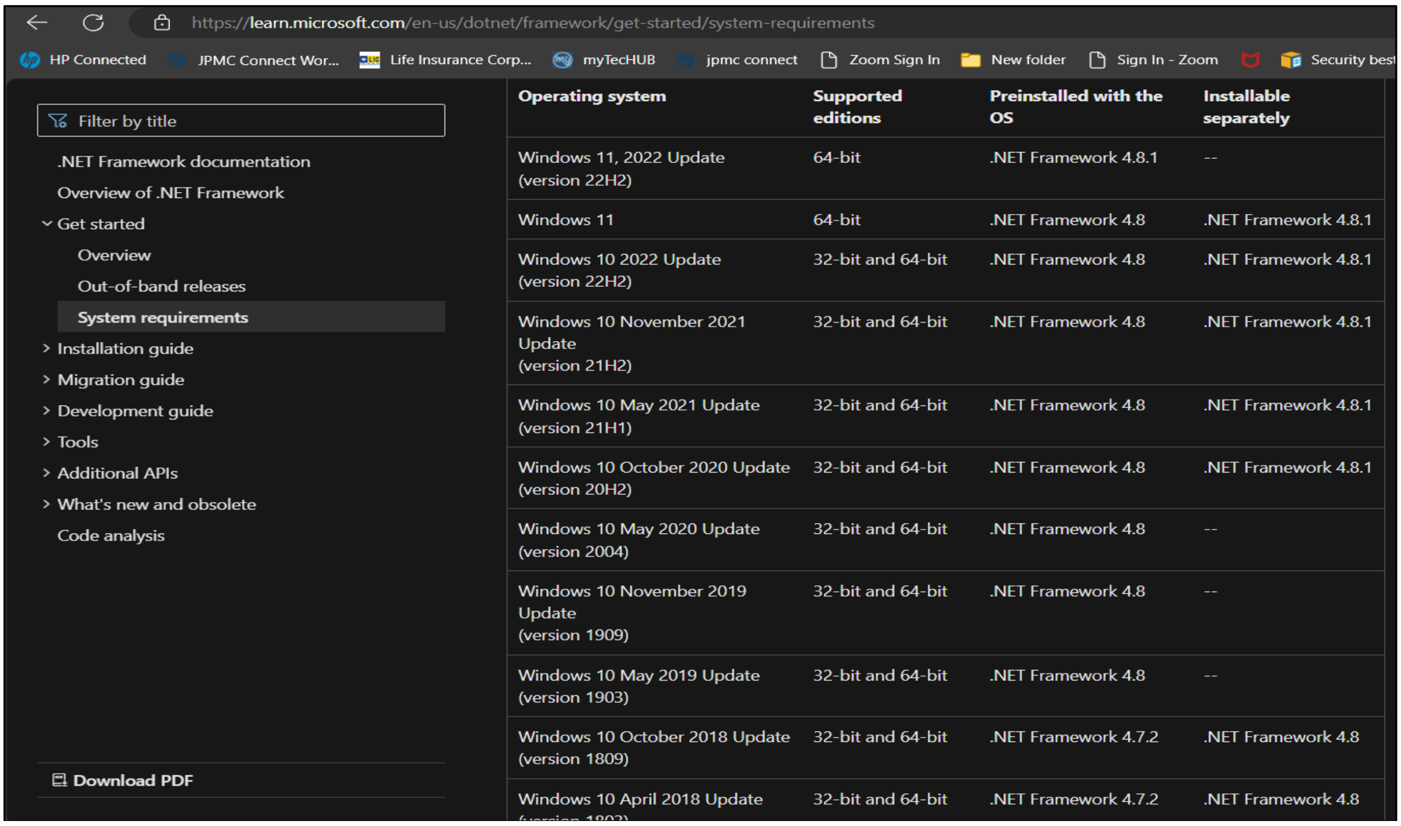
A. Approach

The above-mentioned data is considered as training data on which machine learning models are applied. Consequently, a system upgrade or replacement recommendation is proposed based on weightage of these factors.

Please see Conceptual Framework section for details.

Here below are a couple of examples showing software version, operating system, client, and server version compatibility.

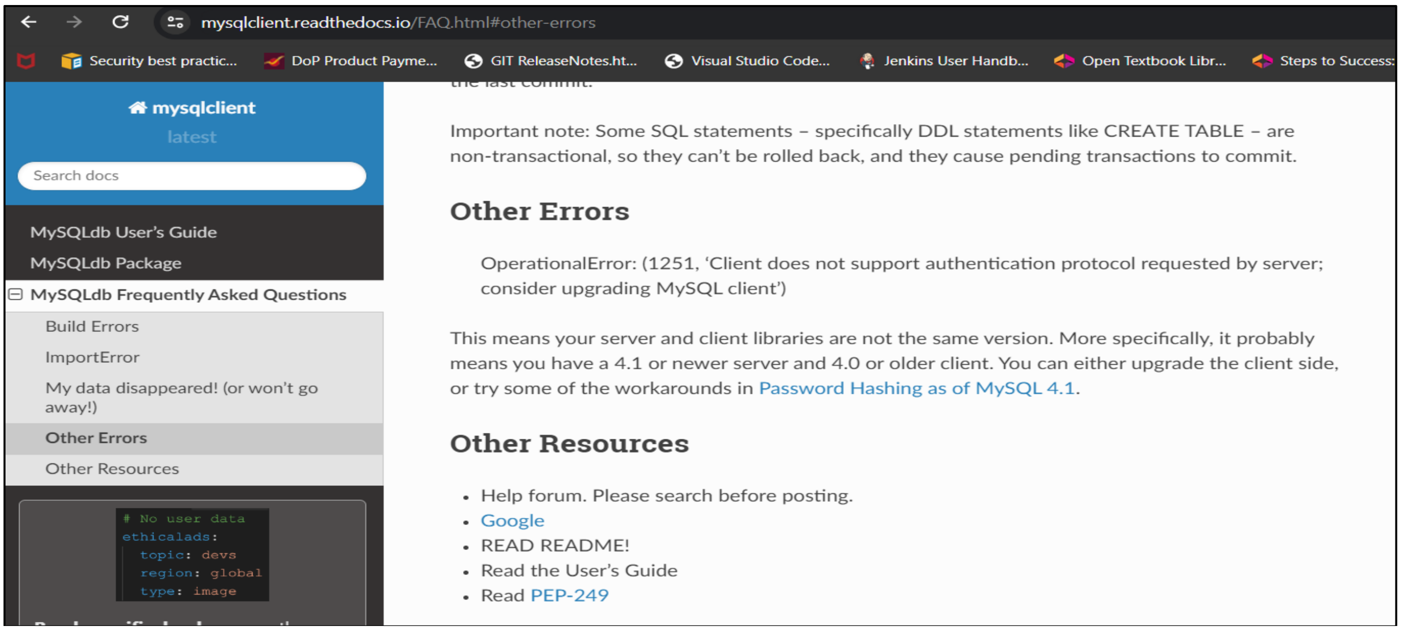
➢ *Operating System Version, .NET Framework Version Compatibility Information:*



Operating system	Supported editions	Preinstalled with the OS	Installable separately
Windows 11, 2022 Update (version 22H2)	64-bit	.NET Framework 4.8.1	--
Windows 11	64-bit	.NET Framework 4.8	.NET Framework 4.8.1
Windows 10 2022 Update (version 22H2)	32-bit and 64-bit	.NET Framework 4.8	.NET Framework 4.8.1
Windows 10 November 2021 Update (version 21H2)	32-bit and 64-bit	.NET Framework 4.8	.NET Framework 4.8.1
Windows 10 May 2021 Update (version 21H1)	32-bit and 64-bit	.NET Framework 4.8	.NET Framework 4.8.1
Windows 10 October 2020 Update (version 20H2)	32-bit and 64-bit	.NET Framework 4.8	.NET Framework 4.8.1
Windows 10 May 2020 Update (version 2004)	32-bit and 64-bit	.NET Framework 4.8	--
Windows 10 November 2019 Update (version 1909)	32-bit and 64-bit	.NET Framework 4.8	--
Windows 10 May 2019 Update (version 1903)	32-bit and 64-bit	.NET Framework 4.8	--
Windows 10 October 2018 Update (version 1809)	32-bit and 64-bit	.NET Framework 4.7.2	.NET Framework 4.8
Windows 10 April 2018 Update	32-bit and 64-bit	.NET Framework 4.7.2	.NET Framework 4.8

Fig 1 Operating System Version

➢ *Mysqldb Library Version Dependencies with Client and Server Versions.*



Important note: Some SQL statements – specifically DDL statements like CREATE TABLE – are non-transactional, so they can't be rolled back, and they cause pending transactions to commit.

Other Errors

OperationalError: (1251, 'Client does not support authentication protocol requested by server; consider upgrading MySQL client')

This means your server and client libraries are not the same version. More specifically, it probably means you have a 4.1 or newer server and 4.0 or older client. You can either upgrade the client side, or try some of the workarounds in Password Hashing as of MySQL 4.1.

Other Resources

- Help forum. Please search before posting.
- Google
- READ README!
- Read the User's Guide
- Read PEP-249

Fig 2 Mysqldb Library Version

B. *Conceptual Framework*

This section reviews various factors in Software or Hardware upgrades. The section begins by describing the uncertainty of when to go for Software or Hardware upgrade and when to eventually retire one or more or consolidate one into another. Further, it has subsections representing dimensions of all possible planning and execution challenges. Each subsection will conclude with a hypothesis that will be used to measure the relationship and dependencies that influence the upgrade or decommission of the Software and associated hardware.

➢ *Systems Asset Documentation*

Systems Asset documentation is a critical bookkeeping activity for any Organization as they are shipped from different vendors and so there are known issues, compatibility gaps between the system assets available vs needed vs used, number of resources needed vs utilized, systems uptime vs downtime, systems idle time vs busy time in Production and Non-Production environments of the Data Centers. The continued monitoring involved here is manual in nature to track what versions are being used, what are being upgraded, what are decommissioned, which code or configurations are obsolete, redundancy factors needed for systems high availability, tracking system alert behavior and patterns, backup and resiliency of system assets, tracking defects and their resolution, and of course reviewing when to go for upgrade or decommissioning of software or hardware.

- *Hypothesis: Asset Data Determines when System Upgrade is Needed.*

➢ *Software and Hardware Compatibility*

Software and Hardware compatibility refers to affinity between software version and associated hardware platform. It is measured by success rate of regular health checks including security scans, execution time, defect resolution turnaround time, system response time after patching or upgrading exercise. With ever increasing demand in software usage along with Artificial Intelligence capabilities and Digital Transformation needs, decisions are taken at the top level and then cascaded to the lower levels. This often leads to improper planning of assets needed to upgrade or decommission because there will be a need of tracking existing issues, open defects and security risks to resolve, tracking end of life components, replacing them with right assets at the right time with minimum down time. So, the level of uncertainty associated with software upgrade or software decommission is usually high when the health of Software and Hardware is not tracked. Hence the need to collect data and metrics associated to assets, on a continued basis.

- *Hypothesis: Software and Hardware Systems Compatibility Influence Systems Upgrade or Systems Decommission.*

➢ *Collecting Metrics*

Collecting metrics is an important task in the software and hardware health check activity. The metrics such as software issues, hardware issues, time taken for regular patches, new issues, security findings, increase in operational cost, increase or decrease in renewal cost, additional upgrade cost, service level objectives w.r.t assets performance, system components to retire etc need to be saved at a centralized location, dependencies to be determined and reviewed periodically.

- *Hypothesis: Software and Hardware System Metrics Help in Taking Timely Decisions in Upgrading, Retiring, Consolidating Assets.*

➢ *Planned Cost*

The cost incurred with software and hardware assets installation and maintenance is another important aspect. The same is used as reference against operational cost to see if there are any deviations. The overall IT expenditure of an organization in a given fiscal year considers this as baseline cost.

- *Hypothesis: Baselined Planned Costs Determine Operational Cost Guidance Year on Year.*

➢ *Actual Cost*

This is the accrued cost in maintenance of software and hardware assets. The overall IT expenditure of an organization comprises of this actual budget spent in the fiscal year against the planned budget guidance start of the year. The profit or loss margins of an organization depend on this important piece of information.

- *Hypothesis: Operational Costs Drive Systems Upgrades and Systems Decommissions.*

➢ *Internal Audit*

The organizations do periodic internal IT system audits of software and hardware components nearing upgrades, end of life, having security risks and vulnerabilities, performance issues to name a few. This is a planning and monitoring exercise where everyone acts according to guidelines defined by the IT Systems head of the department, under the supervision of the Chief Technology Officer and Chief Information Officer. The information tracking needs to be maintained at a certain centralized location, so root cause analysis can take place when things don’t go as expected. Therefore, there is a need to use Machine Learning models to churn the system assets data and system activity data for better decision making considering current and future needs of IT systems.

- *Hypothesis: The IT Audit Findings Drive Systems Upgrade and Decommissioning Need.*

III. RESULTS

This section is divided into four parts: 1) An overview of the research design and the data collection. 2) detailed information regarding the asset data. 3) The technique that will be used to analyze the data. 4) explanation of data privacy and ethical issues that may be associated with the methodology.

➢ *Data Collection Requirements*

This is a blueprint of what data should be collected, and how the data will be analyzed. The design and application of research is dependent upon many factors including the research objective, the availability of the required data source, the cost associated with obtaining the required data, and the time constraints facing the researcher.

➢ *Data Design*

The data design for this research requires both historical and current information about the organization IT assets data. Data for the research is output of weekly exercises on IT Systems maintenance, consolidated risk and run time logs of a software asset version, open resource defects, security vulnerabilities as reported in OWASP, Open-Source Scan vulnerabilities through SNYK etc.

➢ *Data Analysis*

There are two main steps involved in data analysis. They are data preparation, and descriptive statistics. Data preparation will cover the data collection saved to an excel spreadsheet on the computer. The analysis will be done with Python libraries. After that, descriptive statistics and Factor Analysis will be performed. Below listed data is collected as part of this research. The actual data collected and listed can vary, since they are exhaustive and subjective in nature, primarily depends on the IT Systems in use, third party vendor software catalogue, system update activities tracked in an organization and other related data. All this data will be integrated, and outliers to be identified to come up a with recommendation on software upgrade or software decommission, to validate hypotheses outlined in this research.

Table 1 System Asset Master Data

Column	Description	Relation
System_asset_name	Software or hardware system component name	One to one with software or hardware system
Asset_Version	Version of the software or hardware system asset	One to one with system asset
Asset_EOL_date	This is software or hardware expiry date	One to one with system asset
Supported_platform	This denotes operating system, on-premises or cloud specification	One to one with system asset

(*Surrogate Keys are not Defined)

Table 2 System Asset Mapping Data

Column	Description	Relation
System_asset_name	Software or hardware system component name	One to one with software or hardware system
Dependent_asset_name	This denotes the dependent software, hardware, or operating system (on-premises or cloud) specification	One to one with system asset

(*Surrogate Keys are not Defined)

Table 3 System Budget Master Data

Column	Description	Relation
System_version	Software or hardware system component version	Many to one with software or hardware system asset
License_name	Specification of license i.e., enterprise single user, multiuser, single instance, multi instance, operating system association etc.	One to one with software or hardware system version
Planned_cost	This is software or hardware version cost when purchased, deployed	One to one with software or hardware system version
Actual_cost	This is software or hardware version cost when accrued/invoiced	One to one with software or hardware system version
Operation_cost	This is software or hardware version cost when accrued/invoiced year on year or at periodic intervals as applicable	One to one with software or hardware system version
Upgrade_cost	This is software or hardware version upgrade when accrued/invoiced year on year or at periodic intervals as applicable	One to one with software or hardware system version
Decommission_cost	This is software or hardware version decommission cost when accrued/invoiced year on year or at periodic intervals as applicable	One to one with software or hardware system version

(*Surrogate Keys are not Defined)

Table 4 System Activity Master Data

Column	Description	Relation
System_version	Software or hardware system component version	Many to one with software or hardware system asset
Activity_date	This is software or hardware version used	Many to one with software or hardware system version
Activity_code	Activity code description like PATCH UPDATE, RESTART/REBOOT, DOWNTIME etc.	Many to one with activity date
Activity time	Time taken for the maintenance task as mentioned in the activity	Many to one with activity date

	code	
Upgraded_version	Version info of the system if upgraded	Many to one with activity date
Next_activity_date	Next maintenance activity date of the software or hardware version	one to one with activity date
System_asset_sla_passed	1 or 0 representing pass or failed	one to one with activity date
Additional_cost_incurred	Additional cost incurred if any software or hardware failures and replaced with other recommended software or hardware entities	one to one with activity date
Known_issue_count	This is collected from the system errors, warnings or defects encountered from previous activities, or from day-to-day operations tracker	one to one with software or hardware version

(*Surrogate Keys are not Defined)

Table 5 System State Metrics

Column	Description	Relation
System_name	Software or hardware system component name	One to one with software or hardware system
system_version	This is software or hardware version used	One to many with software or hardware system
dep_sw_cnt	Software count on which a software is dependent on	One to many with software version
dep_hw_cnt	Hardware count on which a software is dependent on	One to many with software version
eol_hw_cnt	End of life hardware count associated to software	Many to one with software entity
eol_sw_cnt	End of life software count associated to hardware	Many to one with hardware entity
sw_eol_upg_cost_reqd	This is boolean flag representing if additional cost needed to upgrade the end-of-life software	Many to one with software entity
hw_eol_upg_cost_reqd	This is boolean flag representing if additional cost needed to upgrade the end-of-life hardware	Many to one with hardware entity
hw_maint_cost_reqd	This is boolean flag representing if additional cost needed to maintain/ operate the end-of-life hardware	Many to one with hardware entity
sw_maint_cost_reqd	This is boolean flag representing if additional cost needed to maintain/ operate the end-of-life hardware	Many to one with hardware entity
sw_defects_cnt	The defects count with software version used	Many to one with software version
hw_defects_cnt	The defects count with hardware version used	Many to one with hardware version
hw_min_sla	Minimum service level agreement time in milli seconds for the hardware availability (up and running)	One to one with hardware entity
sw_min_sla	Minimum service level agreement time in milli seconds for the software availability (up and running)	One to one with software entity
hw_upg_recommend	Boolean flag to represent if hardware upgrade needed	One to one with hardware entity
hw_decom_recommend	Boolean flag to represent if hardware decommission is needed	One to one with hardware entity
sw_upg_recommend	Boolean flag to represent if software upgrade needed	One to one with software entity
sw_decom_recommend	Boolean flag to represent if software decommission is needed	One to one with software entity

(*Surrogate Keys are not Defined)

➢ *Data Privacy*

Ethics, as used in research, refers to the expected code of conduct or norms that governs the researcher's behavior while doing research. In this research process, the organizational data privacy will be protected. This research ensures that the information collected from organizations and software products will not be made available to everyone but to the research community. Additionally, all sources that will be used in this research will be duly acknowledged.

IV. CONCLUSION

This Research proposal describes activities in the Software Development Upgrades Decommissions Life Cycle (SDUDLC), establishes relationships, need of data integration to arrive at a decision making on whether Systems need upgrade or decommissions in a timely manner. This systematic data mapping helps in defining the dependencies, needs, priorities, likelihoods of success and failure in the overall process with the introduction of Data Mining and Machine Learning techniques. The below diagram explains the gist of this.

➤ *Proposed Process Flow Diagram of Software Upgrades or Decommissions Life Cycle:*

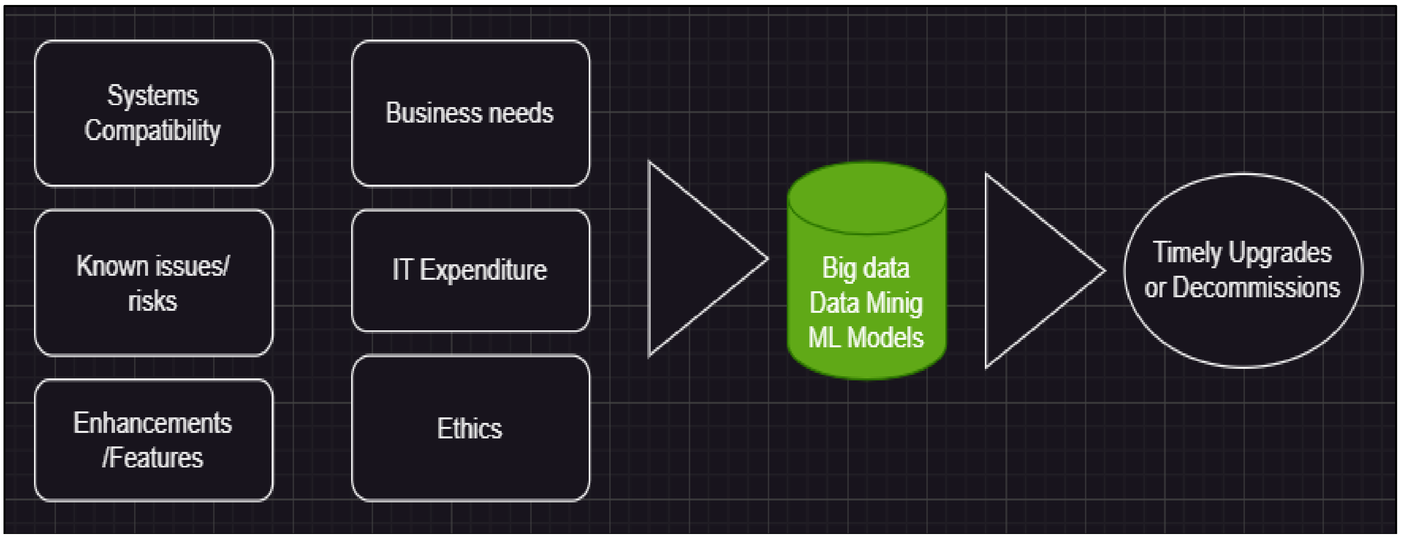


Fig 3 Proposed Process Flow Diagram of Software Upgrades or Decommissions Life Cycle

REFERENCES

[1]. Wikipedia, Free Encyclopedia, https://en.wikipedia.org/wiki/Systems_development_life_cycle
[2]. Fadi Nouh, 2016. SAM Software Asset Management: ResearchGate publication.
[3]. Rekha Bachwani. Olivier Crameri. Ricardo Bianchini and others, 2012. Recommendation system for software upgrades: ResearchGate publication.
[4]. Mauricio Ortiz-Ochoa, 2016. Identifying and Prioritizing Modernization of Legacy Systems: ResearchGate publication.
[5]. Harco Leslie Hendric Spits Warnars and 6 others, 2017. Software metrics for fault prediction using machine learning approaches: ResearchGate publication.
[6]. Intertech, 2022. Software feasibility study: Intertech.com services
[7]. Investopedia, 2022. Feasibility Study Importance: Investopedia.com feasibility-study business essentials publication
[8]. Stefan Van Der Zijden, 2022. Gartner: Three key tasks needed to decommission applications: ComputerWeekly.com publication.
[9]. Ajay Kumar and Kamaldeep Kaur, 2022. Recommendation of Regression Techniques for Software Maintainability Prediction with Multi-Criteria Decision-Making: ResearchGate publication.
[10]. Denis Pombriant, 2021. Do you have the right software for your digital transformation: Harvard Business Review.
[11]. Shahid Iqbal, Muhammad Khalid and M.N.A. Khan, 2013. A Distinctive Suite of Performance Metrics for Software Design: ResearchGate publication
[12]. Marko Saarela, Shohreh Hosseinzadeh, Sami Hyrynsalmi and Ville Leppänen, 2017. Measuring Software Security from the Design of Software: ResearchGate publication.