"GENERATING RECOMMENDATION INSIGHTS AS PART OF THE SOFTWARE UPGRADE AND DECOMMISSIONS LIFE CYCLE IN INFORMATION TECHNOLOGY INDUSTRY"

Research Paper

Ravikanth Kowdeed, Swiss School of Business Management, Geneva, ravikanth@ssbm.ch

"Abstract"

As part of Digital Transformation needs, the Organizations are investing more and more in the Technology and Infrastructure like software upgrades, software renewals, software replacements, Cloud 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 available today which integrates key system assets data such as Software Versions, Platform Compatibility, Dependent Software versions, Investment and Operational Costs, Open defects and fixes, Software Performance Metrics and Service level objectives. Due to this, the so-called decision making is a tedious process that takes time and effort. This research paper proposes Software Upgrade and Decommissions Life Cycle and outlines the requirements, design and build approach for generating recommendation insights. This also proposes using Data Mining and Machine Learning models on the input data sets that are needed to take a decision on software upgrade or software decommissioning.

1. Introduction

This research paper proposes Software Upgrades and Decommissions Life Cycle as a continuous process in which various data sets are collected, integrated so Data Mining and Machine Learning models can be applied for generating valuable insights. The methodology considered here is based on case studies in systems modernization (Mauricio, 2016) and exploring software release documentation available since the year 2000. The goal is to gather various data sets needed, apply Pareto law that states - 80% of consequences come from 20% of causes. The data model is defined to establish relationship between the system asset data sets (Fadi, 2016), helps to perform data exploratory analysis and apply Machine learning models on this big data. This further helps in fault prediction (Harco et al 2016; Ajay, Kamaldeep, 2022) and generating data insights that shows influence of Software asset versions, version dependencies, platform compatibility, performance metrics (Shaid et al, 2013), hardware dependencies, defects and vulnerabilities in the decision making of Software upgrades (Rekha et al, 2016) or software replacements. Overall, the recommendation insights derived from the below-mentioned data sets, help expedite decision making of software upgrades or decommissions (Stephan, 2022) in information technology field to meet today's Digital Transformation needs (Denis, 2021).

1. Software Requirements (Business objective, System needs, Software features, Hardware

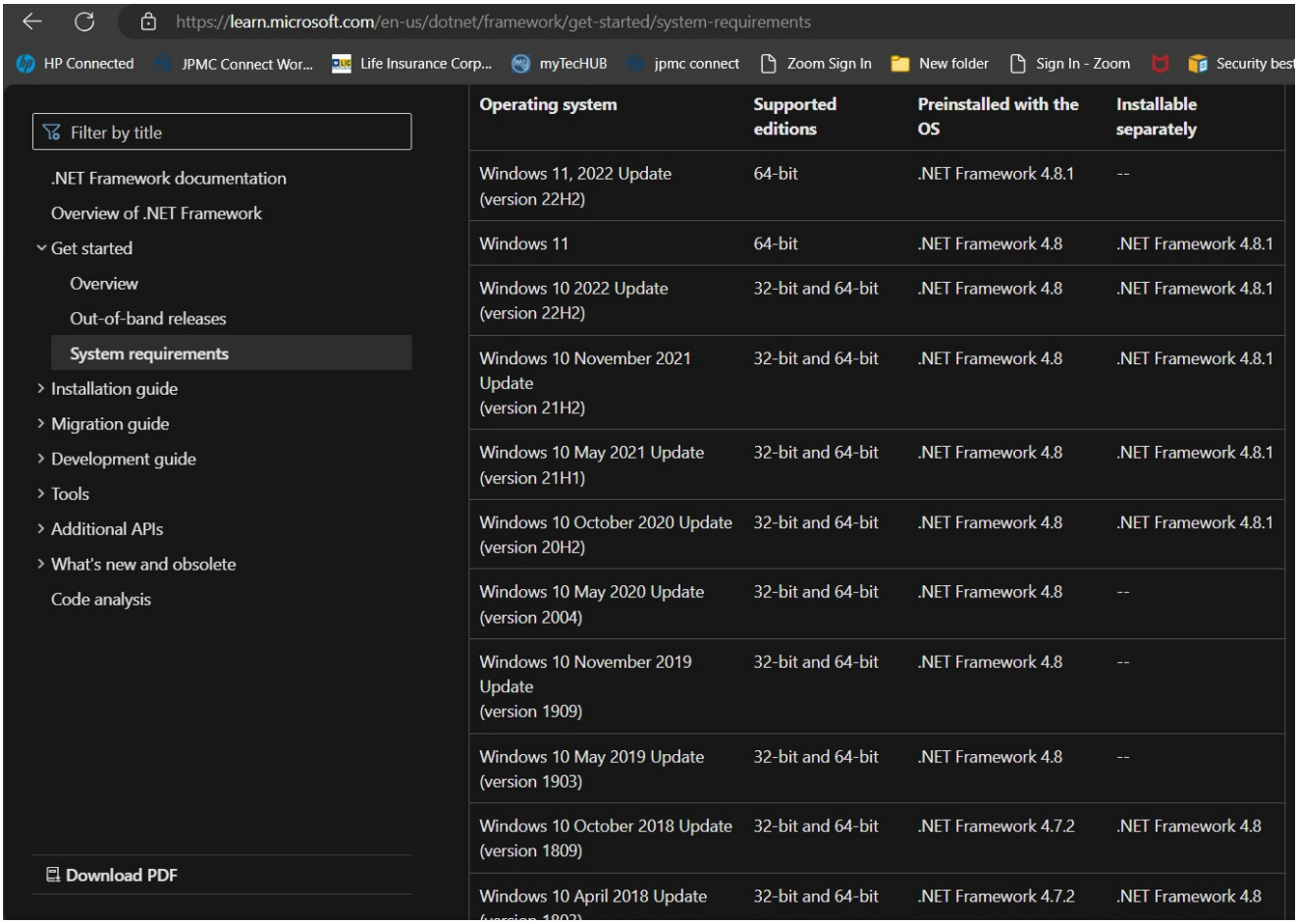
specifications, Cloud vs non-Cloud infrastructure supported).
2. Software Cost (Invest vs Operate Cost i.e., installation, renewals, upgradation and decommission costs)
3. System Asset Metadata (Software features, version information, service level objective, end of life date).
4. Software Issues (Compatibility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

2. Research Methodology

The publicly available data will be loaded and integrated for data analysis, after which the machine learning models are trained. Consequently, a system upgrade or replacement recommendation is proposed based on weightage of these factors.

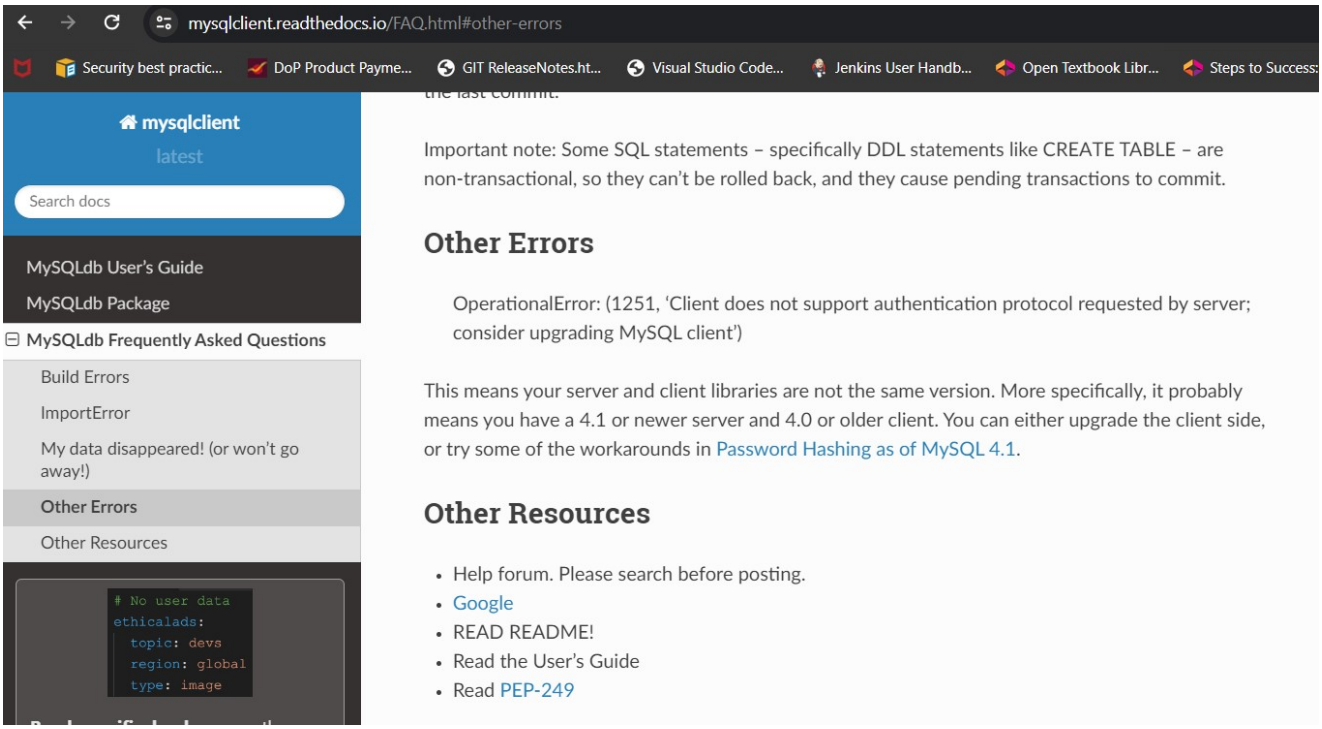
Here below are a couple of examples showing software version, operating system, client, and server version compatibility.

Operating system version, .NET framework version compatibility information is shown in Fig.1. below.



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

MySQL dB library version dependencies with client and server versions shown in Fig.2. below



More such data is pulled from publicly available release documentation to review and consider various factors in the process. The goal here is to answer when to go for Software or Hardware upgrade and when to eventually retire one or more or consolidate one into another. Further, this methodology describes subsections representing dimensions of all possible planning and execution challenges. Each subsection concludes with a hypothesis that is used to measure the relationship and dependencies that influence the upgrade or decommission of the Software and associated hardware.

2.1 Systems asset documentation

Systems Asset documentation is a critical bookkeeping activity for any Organization. The system components either software or hardware get 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 which versions are being used, what needs upgrade, what needs replacement and then decommissions, which code or configurations are obsolete, need for systems high availability, system alerts, backup and resiliency events, tracking defects and their resolution, and of course cost associated with upgrade or decommissioning of a software or hardware.

Hypothesis: asset data determines when system upgrade is needed.

2.2 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.

2.3 System 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 (Mark et al, 2017), increase in operational cost, increase or decrease in renewal cost, additional upgrade cost, service level objective misses, tracking end of life for system components 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.

2.4 Planned and actual costs

The planned cost is incurred cost with software and hardware assets installation and maintenance. The overall IT expenditure of an organization on a quarterly basis or in a given fiscal year considers this as baseline cost. The accrued cost is the actual cost incurred in the financial year on specific software and hardware assets together. The actual and planned costs are compared periodically to see if there are any deviations. The overall IT expenditure of an organization comprises of the actual budget spent in the fiscal year against the planned budget forecast start of the year. The profit or loss margins of an organization depend on this important piece of information

Hypothesis: baselined planned costs determine operational cost guidance year on year. Operational costs drive systems upgrades and systems decommissions.

2.5 Internal guidelines and procedures

The organizations undergo periodic system audits of software and hardware components requiring upgrades, replacing end of life ones, monitoring security risks and vulnerabilities, addressing performance issues to name a few. This is a regular planning and monitoring exercise following the

guidelines defined by the head of the IT Systems department, under the supervision of the Chief Technology Officer and Chief Information Officer. All this information thus tracked needs to be maintained at a centralized location, so root cause analysis is done 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.

3. 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 are associated with the methodology.

3.1 Data collection

This is a blueprint of what data should be collected, and how the data will be analyzed. The design and application of research depends on many factors including the research questions, scope and the availability of the required data source. While most of the information is publicly available, sometimes there is cost associated with obtaining some data as there are licensed tools and products in the market. This adds time constraint to the problem. Thankfully, ChatGPT is trained on large data sets and can be leveraged to extract historical data in the research. For data after 2021 September, the release documentation of the software products can be leveraged for data collection.

3.2 Data design

The data design for this research requires both historical and current information about the organization IT assets data. All this data plus Systems activity data collected periodically, consolidated risk metrics, security defects and vulnerabilities as reported in OWASP, SNYK, operational cost etc are considered. The data is modelled to load in Relational Database, as below.

System Asset Master Data

(*surrogate keys are not defined)

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

System Asset Mapping Data

(*surrogate keys are not defined)

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

System Budget Master Data

(*surrogate keys are not defined)

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

System Activity Master Data

(*surrogate keys are not defined)

Column	Description	Relation
System_version	Software or hardware system	Many to one with

	component version	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 code	Many to one with activity date
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

System State Metrics

(*surrogate keys are not defined)

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

3.3 Data ethics and copyrights

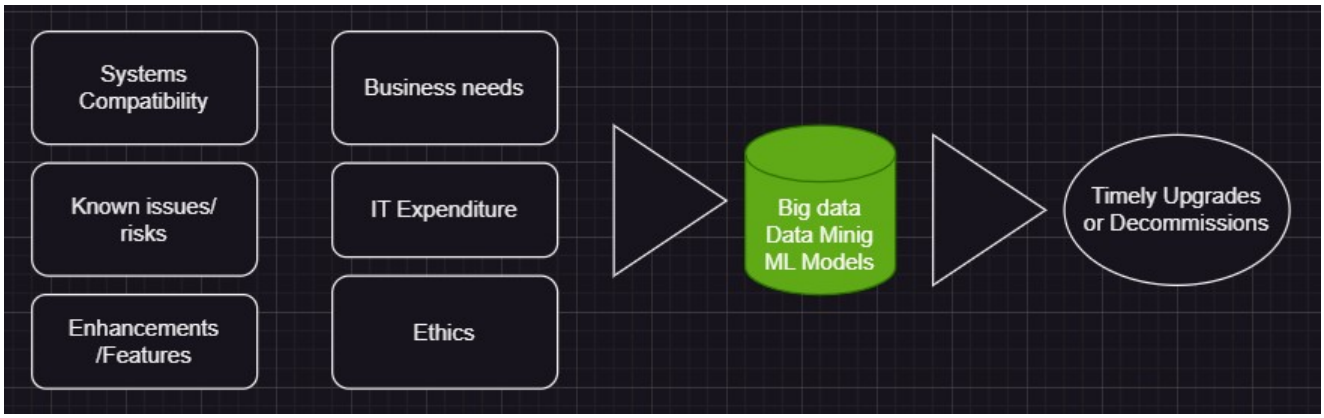
Ethics, as used in research, refers to the expected code of conduct or norms that governs the researcher's behavior while doing research. In this research process, the organizational data privacy and overall proprietary data boundaries will be respected. This research ensures the copyrighted information collected from organizations and software products will not be disclosed, only the case studies and release documentation that are publicly available is used. Additionally, all sources that will be used in this research are acknowledged with reference or screenshots.

3.4 Data mining

The data analysis comprises of data exploration and applying descriptive statistics. Data collected is saved to excel spreadsheets on the computer and then loaded to Relational Database for data mining activity. Python programs with appropriate libraries are used to produce descriptive statistics and Factor Analysis on the collected data. The actual data collected and baseline data available 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 and other related data tracked every week or monthly basis in Organizations. All the input data will be integrated, and outliers to be identified periodically to produce recommendation insights on software upgrades or replacements needed on a proactive basis. All this helps to validate hypotheses outlined in this research.

4. Conclusion

With the evolution of Generative Aritificial Intelligence and Large Language Models, the process of feeding essential information and data retrieval is getting simplified, but these are still evolving as there is a need of private LLMs to be used for this use case because some of the Organizational data may have copyrights. Hence, exploratory data analysis and applying machine learning models like Time Series, Random Forest, K-NN, SVM, ARIMA are used for fault predictions. These are helpful in deriving key insights and recommendations, as part of the proposed process of Software Upgrades or Decommissions Life Cycle. Here is the flow diagram:



References

Bachwani R., et al (2012) 'Recommendation system for software upgrades', *ResearchGate publication* [online]. Available at https://www.researchgate.net/publication/234128761_Mojave_A_Recommendation_System_for_Software_Upgrades

Iqbal M., Khalid M. and Khan M.N.A. (2013) 'A Distinctive Suite of Performance Metrics for Software Design', *ResearchGate publication* [online]. Available at https://www.researchgate.net/publication/270526905_A_Distinctive_Suite_of_Performance_Metrics_for_Software_Design

Kumar K. and Kaur K. (2022) 'Recommendation of Regression Techniques for Software Maintainability Prediction with Multi-Criteria Decision-Making', *ResearchGate publication* [online]. Available at https://www.researchgate.net/publication/362917558_Recommendation_of_regression_techniques_for_software_maintainability_prediction_with_multi-criteria_decision_making

Nouh F. (2016) 'SAM Software Asset Management', *ResearchGate publication* [online]. Available at https://www.researchgate.net/publication/291818013_SAM_Software_Asset_Management

Ortiz-Ochoa M. (2016) 'Identifying and Prioritizing Modernization of Legacy Systems', *ResearchGate publication* [online]. Available at https://www.researchgate.net/publication/294874894_A_Practical_Approach_to_Identifying_and_Prioritizing_Modernization_of_Legacy_Systems

Pombriant D. (2021) 'Do you have the right software for your digital transformation', *Harvard Business Review, 2021(8)*. Available at https://hbr.org/2021/08/do-you-have-the-right-software-for-your-digital-transformation

Saarela M., et al (2017) 'Measuring Software Security from the Design of Software', *ResearchGate publication* [online]. Available at https://www.researchgate.net/publication/321140241_Measuring_Software_Security_from_the_Design_of_Software

Singh M. and Chabbra J.K. (2021) 'Software Fault Prediction Using Machine Learning Models and Comparative Analysis', *ResearchGate publication* [online]. Available at https://www.researchgate.net/publication/350490953_Software_Fault_Prediction_Using_Machine_Learning_Models_and_Comparative_Analysis

Zijden S.V.D. (2022) 'Three key tasks needed to decommission applications'*, Computer Weekly article*, [online]. Available at https://www.computerweekly.com/opinion/Gartner-Three-key-tasks-needed-to-decommission-applications .