International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 10 October 2026
Page No: 5796 - 5756
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0752

A Technical Debt as Institutional Memory- A Narrative Review of Legacy Modernisation in Regulated Financial Systems

Swaminathan Vaidyanathan[1]
[1]*Sri Chandrasekharendra Saraswathi Viswa Mahavidyalaya (SCSVMV) Enathur, Tamil Nadu 631561, India*
Email id: *swamivaith@gmail.com*[1]

Abstract
Technical debt is usually measured based on its impact on software quality, maintenance work and change. For regulated financial systems, however, a legacy implementation may also have business rules, the need for which may not be well documented. This is a narrative review that is a synthesis of literature on technical debt, software evolution, legacy modernisation and organisational memory, banking guidance and a documented operational incident, all of which are used to investigate that tension. It suggests the debt-as-memory hypothesis: there are technically undesirable artefacts that contain information that is needed to interpret and/or reproduce an institutional obligation. The synthesis identifies three markers of these artefacts: uncertain rationale, lack of independent evidence and consequences that occur beyond the immediate change process. A conceptual taxonomy is provided that links the candidate debt forms to the knowledge at risk and a comparison of six modernisation strategies is provided that outlines their potential for retaining the behaviours and the evidentiary requirements for each. Practical assessment is communicated through: Provenance review, examination of historical exception, self-documented proven rules, and test of planned behaviour. The analysis does not seek to measure the amount of memory bearing debt, nor to prioritise the modernisation strategies. It also does not have any justification for a definite 10-year limit after which documentation is no longer reliable. The review may consider the meaning of the legacy behaviour and then decide if it should be retained, amended or phased out.

Keywords: *technical debt; legacy modernisation; institutional memory; regulated financial systems; software evolution.*

1. Introduction
Technical debt is a metaphor for the future cost of continuing to use a software design with incomplete understanding [1] introduced by Cunningham. The idea was further refined into identification, classification and management strategies for compromises which make it more difficult to change later [2,3]. The reason why repeated adaptation can cause a system to become more complex and make the correspondence between the system and its documentation weaker is further explained by software evolution and ageing research [4, 5]. These views set the maintenance problem, and raise a question of information: What knowledge might be lost when a legacy component is removed? One of the reasons why legacy systems are difficult to replace is that they are tied into business processes and dependencies that are not well understood [6, 7]. In the present review, the authors investigate whether there is some form of institutional decision that can be preserved by some apparent technical debt. Building an application just from its current specification can fail to capture behaviour that was still in production code but not documented in the specification. This is offered as a particular mechanism of knowledge loss and not a general explanation of unsuccessful modernisation. One such documented case is the RBS Group incident in June 2012. Staff rolled back a patched version of the batch scheduler, without testing the rollback of that specific version, during an upgrade of the NatWest and Ulster Bank batch scheduler. Then jobs were not appearing in the processing queues, causing the sequence to be broken to update customer accounts. The FCA also noted incomplete change records and a lack of documentation of previous rollback actions [8, paras. 4.6–4.11, 4.23 and 4.42]. This was an upgrade/recovery failure,

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 10 October 2026
Page No: 5796 - 5804
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0752

not a documented deletion of a contractual rule. However, it is a good example of how technical change can render the knowledge of the system state and dependencies inaccessible to those tasked with continuity. The present paper takes as its analytical interpretation of the incident the knowledge as institutional memory. The business-rule mechanism is shown with a hypothetical fee exemption. If the legacy routine omits the accounts opened prior to the merge, and the replacement specification does not include the standard fee, then the accounts will not be merged. Taking that branch down would also take down the executable expression of the historical entitlement, unless the rationale and affected population had been documented elsewhere. It is a representative example of the literature that acknowledges that legacy systems hold organisational knowledge within them, not a banking case or an observed outcome. [9] The review combines technical debt and organisational memory to separate the structural maintenance issues from the information contained in them. It creates a conceptual taxonomy of potentially memory-bearing artefacts and a comparison of six modernisation strategies with respect to the retention of behaviour and the evidentiary requirements. The supporting literature is explored in the following sections, the narrative synthesis is explained, assessment criteria are developed and implications and limitations are discussed.

2. Background and Related Work

2.1.From metaphor to management

Technical debt research reviews encompass the identification, measurement and management aspects [3, 10]. Tom and colleagues investigated the dimensions and boundaries of the concept [11], and research on architectural debt studies the effects of design choices and the challenges of assessing them using local code measures [12, 13, 14]. Here, “practitioners” refers to software engineers, architects, and technical managers, and is not related to teaching practice. In the study, Ernst and colleagues interviewed seven engineers and surveyed 1,831 participants, concluding that architectural decisions were a major source of debt and that tools offered limited support for managing the details of the debt [15]. Measurement approaches also account for remediation costs and extra effort from deferred work [16]. The memory view brings another question into play: is this technically bad implementation also the only trace of the business rule?

2.2.Legacy modernisation

Legacy modernisation literature considers existing systems to be business assets that must be replaced, taking into account technical constraints as well as operational continuity [6]. Comella-Dorda et al. discuss methods such as wrapping and integration and componentisation [9]. Seacord and colleagues discuss the engineering and business processes of modernisation [17] and Sneed discusses planning decisions that relate to re-engineering [18]. The testing technique of Feathers, and Fowler's refactoring methods are complementary ways to modify existing code while managing behaviour changes [19, 20]. Practitioner research reveals that in addition to technical obsolescence, knowledge and organisational issues are important concerns [7]. Brooks' taxonomy of essential and accidental complexity is a helpful lens: an awkward implementation may be a necessary rule, but not every form of complexity is necessary.

2.3.Organisational memory

According to Walsh and Ungson, organisational memory is the information that is carried over from an organisation's past that is relevant to decisions made in the present [22]. Nelson and Winter focus on routines as the core of the knowledge reproduction of organisations [23]. This review brings those perspectives to bear on executable software. A program can reproduce a rule after those who established it have gone away, even if the reasons for the rule are no longer available. The argument is not about whether software is excluded from earlier theories of organisational memory, or whether all operational routines have a permanent validity.

2.4.The regulatory amplifier

The integrity of operational information is especially critical in the context of banking guidance. BCBS 239 sets standards for the aggregation and reporting of risk data that are accurate, complete and timely [24]. The Basel

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 10 October 2026
Page No: 5796 - 5804
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0752

Committee's operational resilience principles also focus on mapping of the dependencies required to provide critical operations [25]. These principles are to be used as a governance framework for the investigation of unexplained calculations and dependencies and are not a definition of "memory-bearing debt". The review infers that maintaining a rule without knowledge of its origins may create an evidentiary void, even if the existing system yields consistent outcomes.

3. Review approach

This paper is not an empirical analysis of a particular set of modernisation engagements, but a narrative review with a conceptual synthesis. It brings together the referenced literature on technical debt, legacy systems and organisational memory and banking guidance and a documented regulatory case. However, the interpretive contribution of narrative synthesis can be used to explore links between research traditions with disclosure of the limits of such synthesis [26]. No registered protocol, exhaustive database search, formal screening flow or quantitative pooling is claimed. The author's claim of two-and-a-half decades of professional experience is used to provide motivation rather than a sample for the study. Four application areas are discussed: core banking, payments, insurance policy administration and regulatory reporting. The comparison in Table 2 is structured in six categories of strategy, based on known modernisation approaches [9, 17, 18]. These categories can overlap in one programme and should not be seen as six engagements. Due to the lack of a verified engagement register with the review, an approximate number of projects analysed cannot be provided; to assign a number would be misleading as it would not reflect the evidence base. The review differentiates between published findings and documented incidents and illustrative scenarios. The RBS Group incident is an externally recorded case of change and recovery failures [8]. The examples of fee-exemption and the calendar, rounding and reporting scenarios in the tables illustrate how the proposed mechanism could work. They are conceptual drawings based on modernisation and legacy-testing literature, rather than further observations or estimates of failure frequency [17, 19]. The synthesis relates the maintenance implications of technical debt to the retention of organisational knowledge, and then considers how different types of technical change could affect the retention of organisational knowledge [2, 22, 23]. The resulting conceptual taxonomy is expressed in Table 1 and strategies are compared in Table 2, based on qualitative assumptions about retained behaviour and evidence required to justify change. Analytical aids proposed are discovery intervals, ratings and practical criteria. These are not pooled results, validated scores or estimates from an engagement cohort.

4. The Debt-as-Memory Hypothesis

The debt-as-memory hypothesis suggests that an implementation can impose maintenance costs, but keep information pertinent to an existing obligation. This interpretation is a synthesis of the literature on technical compromises and the perspective that organisational routines are a way to store knowledge [2, 22, 23]. Referential opacity is when the implementation shows what is happening, but not why. For instance, a conditional can be that the customer population is excluded, but it doesn't say why. That behaviour is not necessarily needed, and neither code complexity nor the lack of comments is enough. Evidentiary singularity is a practical finding that no authoritative explanation has been found independent of the other, having examined the available records. It is a conclusion drawn from the evidence presented to the person reviewing the code, not an attribute that can be directly measured in the source code. Asymmetric loss refers to a disconnect between the decision to change and the subsequent realization of the impact of the change, e.g., a mismatch in the next reporting period. These concepts are an extension of the organisational memory perspective from the review to the traceability and dependency issues identified in banking guidance [22, 24, 25]. The discovery points shown in Table 1 are not measured discovery times. Assessment should start by determining if the component has an impact on a customer entitlement, contractual calculation, settlement convention, reporting output, retention duty or

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 10 October 2026
Page No: 5796 - 5804
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0752

operational control. The engineer and responsible business owner should then review to determine if the reasoning is known and if there is an independent record to support the reasoning. Examples of relevant evidence include approved specifications, change records, version history and control documentation. This is done by mapping the governance and dependency issues in the banking principles mentioned above to an evaluation of the particular artefact [24, 25]. The assessment should also take into account the location of the discrepancy and whether the tests available would cover the discrepancy. Historical transactions, dormant accounts or period-end processing may be used infrequently and not be covered by the recent high-volume data. Legacy-testing techniques assist in creating explicit tests on the basis of already existing behaviour, and the results of the RBS show the impact of testing a rollback without any representative conditions [8, 19]. The practical record can therefore be used to record the relevant obligation and rationale, and to record in continuous narrative form, with links to supporting records, the likely exposure point and test coverage. The criterion offered by the review is that a component should be studied further if it has an impact on an obligation, and either the rationale for the obligation is not discussed or there is no independent evidence of the component. The criterion is an interpretive screening tool, rather than a validated score or regulatory requirement. The decision as to whether the legacy treatment is still appropriate is left to the business owner and behaviour that has been coded is not necessarily correct. Through traceable decisions and tests [19, 20] the approved corrections should be distinguished from the unintended differences.

Table 1 Conceptual taxonomy of potentially memory-bearing technical debt

Observed debt form	Memory it encodes	What is lost on blind removal	Illustrative discovery point (not measured)
Undocumented conditional branch guarding a customer or product segment	A negotiated exemption, grandfathering clause or supervisory waiver	Population-level misclassification; ineligible accounts treated under current rules	One reporting cycle to several years
Hard-coded date thresholds and bespoke calendar arithmetic	Settlement conventions, regime transition dates, holiday regimes at a point in time	Silent value-date and accrual errors	Month-end reconciliation
Non-standard rounding	Agreements with auditors on materiality and tie-breaking	Sub-cent divergence reconciliation	Immediate to one cycle, but hard to attribute
Duplicated calculation logic across channels	Deliberate divergence between statutory, management and customer-facing figures	Spurious harmonisation that destroys a required distinction	External audit or regulatory review

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 10 October 2026
Page No: 5796 - 5804
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0752

Dormant fields, unused columns and defensive null handling	Discontinued products still bearing retention or restitution obligations	Inability to answer look-back, data-subject requests	Years, on regulatory request
Batch sequencing and undocumented job dependencies	Operational risk controls and learned from past incidents	Reintroduction of mitigated failure modes	First stress or peak-volume event

5. Implications of the Review

An undocumented conditional is a conditional that should be investigated because it is technically documented, but not for a business purpose. Legacy system research acknowledges that within the organisation's past there is knowledge that can be kept in legacy systems [9]. This does not mean that all the exceptions have to be made. A good assessment distinguishes between the cost of sustaining the implementation and the evidence for the rule, so that the obsolete logic can be eliminated without the missing documentation being regarded as evidence of obsolescence. There is a gap between documentation and actual behaviour, but there is no evidence that there is a 10-year expiry date. Lethbridge, Singer and Forward found that documentation changes were not uniformly documented in three studies, and that documentation might still be useful despite being obsolete [27]. Hence, a repository's assessment should be based on its association with existing code, its accepted modifications and tests, not just its age. The review does not say that all repositories have a similar deterioration curve, but does suggest that undocumented fixes and changing product rules are potential paths to divergence. Production code should not automatically take precedence over documentation if they conflict. Calculation and reporting boundaries are especially important to examine since the significance of an output is dependent on how it is classified, its scope and intended use. BCBS 239 gives a foundation to discuss accuracy and completeness of risk information [24]. If this is applied to modernisation it means that seemingly redundant calculations must be compared at the level of their business purpose before they are consolidated. While it is valid for a statutory figure and internal management figure to be different, the difference must be provided by the relevant reporting requirements and not by the fact that the figures are reported separately in the code. Behavioural comparison can be helpful if a specification is not fully complete. Characterisation tests can log the behavior of legacy code against chosen inputs, which can be used to measure the behavior of subsequent changes [19]. In the application of the approach in this review, comparison should cover relevant boundary conditions and low-frequency business events. The differences need to be interpreted: they can be replacement defects or environmental changes or approved changes in the legacy system. The untested patched rollback in the RBS notice is a good example of the importance of evidence to include the actual change path, not a superficially similar version [8]. Finite tests can only be used to draw conclusions within their scope and not for all possible executions. Debt assessment should also differentiate between maintenance effort and informational dependence. Cost oriented approaches are still helpful in determining the impact of delayed technical work [16] but are not sufficient on their own to determine whether a component still provides the only remaining explanation of an obligation. Based on the organisational memory perspective [22] the review therefore suggests to record the provenance uncertainty together with the remediation cost. The proposed mechanism is summarized in figure 1: once a rule is put in the implementation, the rationale behind it becomes less accessible, and in modernizing the system either the rationale is externalized or there is a risk of losing it. The cycle shown is a conceptual cycle and is not a measured sequence of projects observed. As shown in Figure 1 Conceptual lifecycle of technical debt as

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 10 October 2026
Page No: 5796 - 5804
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0752

institutional memory.

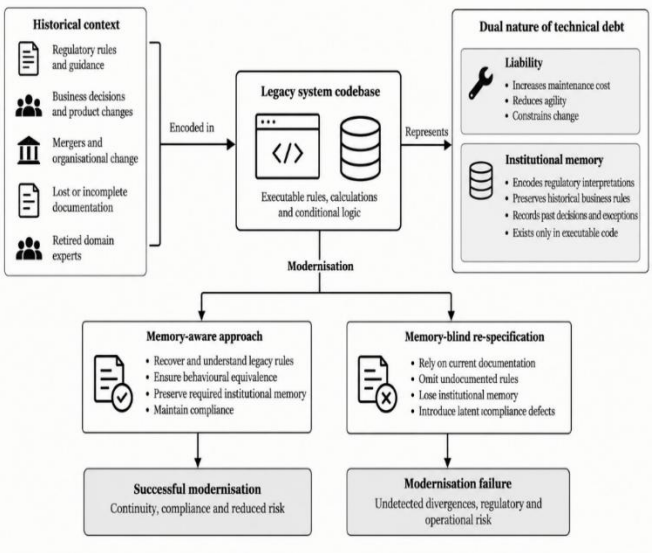

Figure 1 Conceptual lifecycle of technical debt as institutional memory

6. Implications for Memory-Aware Modernisation

The synthesis provides a starting point for modernisation by examining the origin and dependence of the rules. Records can exist in the form of existing records and relevant domain expertise can help identify which behaviour represents a continuing obligation; versioned specifications and executable tests are examples of records that can outlast the original component. So far, controlled behavioural checks during structural change are based on legacy testing and refactoring literature [19, 20]. This is not to say that this is an experimental finding that one sequence is better, but rather a practical consequence of the review. Table 2 is an adaptation of proven modernisation approaches to the two qualitative dimensions of the review [9, 17, 18]. Memory retention is not the preservation of the explanation of the rule behaviour, but the behaviour itself. High retention – original rule implementation continues to execute; moderate retention – translation or selective reconstruction; low retention – rules must be recovered and expressed in a significantly different implementation. Evidentiary burden is the extra analysis and testing required to support the modification of the behaviour or environment. The ratings are the reasoned judgements of the review and do not necessarily reflect performance measurements of the sources cited. These assumptions are stated in the rating examples. Wrapping an unchanged fee routine will have the same behaviour, but will still need to be verified for interface mapping and transaction boundaries. Replatforming preserves source logic with potential variations in numeric accuracy, calendars or order of execution. If the target model doesn't have the local rules, more extensive reconstruction may be needed for a package replacement or rewrite. These illustrations are based on the change-control and testing principles from the modernisation literature [17, 19]. Ordering is conditional: if the change is extensive, more evidence may be required than if it is narrowly scoped and there are complete specifications and tests. As shown in Table 2. Qualitative comparison of modernisation strategies.

Table 2 Qualitative comparison of modernisation strategies

Strategy	Memory retention	Evidentiary burden	Main risk	Suitable context
Service wrapping	High	Low	Wrapper complexity	System integration
Replatforming	High	Moderate	Behavioural changes	Platform replacement
Code transformation	Moderate	High	Translation errors	Language obsolescence

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 10 October 2026
Page No: 5796 - 5804
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0752

Incremental re-engineering	Moderate–high	High	System inconsistency	Gradual replacement
Package replacement	Low	Very high	Rule omission	Standardised functions
Greenfield rewrite	Lowest	Highest	Knowledge loss	Documented requirements

7. Discussion and Limitations

The review does not aim to be comprehensive or to formally evaluate all the relevant literature, but it does link together existing research traditions. Examples and emphasis may be dependent on narrative source selection and author's professional view, and are not easily replicable or broadly applicable [26]. The conceptual taxonomy does not show the level of memory-bearing debt and the strategy comparison does not show comparative safety or cost-effectiveness. It contributes a framework for recognising an evidence problem which can be hidden by a purely structural assessment of software. Further evaluation would need to be based on clearly defined empirical cases with selection and provenance information, change decisions and outcomes. Guidance for case-study reporting provides a basis for that future validation [28]. The implications for the present review are more limited: technical debt needs to be considered from both a maintenance perspective and in terms of the knowledge that may be lost when changes are made. There must be a mechanism for justified correction and retirement as well as retention, but only if the underlying obligation is valid.8.

Conclusion

This is a narrative review that highlights a possible conflict between technical debt reduction and institutional knowledge in regulated financial systems. The debt-as-memory perspective focuses on the murky rationale, the lack of independent evidence and the potential side effects of a technical change that might be accepted. The taxonomy and the comparison of strategies give a foundation for the analysis of these conditions without the need to assign unverified frequencies or universal ratings. Practical assessment shall determine the corresponding obligation, trace its origin and validate its implementation, before discarding its implementation. Documentation age is not a criterion for reliability and previous behaviour should not be maintained without investigation. When decisions are made about what to retain, correct or remove, they are better supported by accessible evidence to help modernisation.

References

[1]. Cunningham, W. (1992). The WyCash portfolio management system. OOPSLA '92 Experience Report. ACM SIGPLAN OOPS Messenger, 4(2), 29-30.
[2]. Kruchten, P., Nord, R. L., & Ozkaya, I. (2012). Technical debt: From metaphor to theory and practice. IEEE Software, 29(6), 18-21.
[3]. Li, Z., Avgeriou, P., & Liang, P. (2015). A systematic mapping study on technical debt and its management. Journal of Systems and Software, 101, 193-220.
[4]. Lehman, M. M. (1980). Programs, life cycles, and laws of software evolution. Proceedings of the IEEE, 68(9), 1060-1076.
[5]. Parnas, D. L. (1994). Software aging. In Proceedings of the 16th International Conference on Software Engineering (ICSE '94) (pp. 279-287). IEEE Computer Society Press.
[6]. Bisbal, J., Lawless, D., Wu, B., & Grimson, J. (1999). Legacy information systems: Issues and directions. IEEE Software, 16(5), 103-111.
[7]. Khadka, R., Batlajery, B. V., Saeidi, A. M.,

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 10 October 2026
Page No: 5796 - 5804
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0752

Jansen, S., & Hage, J. (2014). How do professionals perceive legacy systems and software modernization? In Proceedings of the 36th International Conference on Software Engineering (ICSE '14) (pp. 36-47). ACM.

[8]. Financial Conduct Authority, "Final notice: Royal Bank of Scotland Plc, National Westminster Bank Plc and Ulster Bank Ltd," Nov. 19, 2014, paras. 4.6–4.11, 4.23 and 4.42. [Online]. Available: https://www.fca.org.uk/your-fca/documents/final-notices/2014/rbs-natwest-ulster. Accessed: Oct. 6, 2026.

[9]. Comella-Dorda, S., Wallnau, K., Seacord, R. C., & Robert, J. (2000). A survey of legacy system modernization approaches (Technical Note CMU/SEI-2000-TN-003). Software Engineering Institute, Carnegie Mellon University.

[10]. Alves, N. S. R., Mendes, T. S., de Mendonca, M. G., Spinola, R. O., Shull, F., & Seaman, C. (2016). Identification and management of technical debt: A systematic mapping study. Information and Software Technology, 70, 100-121.

[11]. Tom, E., Aurum, A., & Vidgen, R. (2013). An exploration of technical debt. Journal of Systems and Software, 86(6), 1498-1516.

[12]. Besker, T., Martini, A., & Bosch, J. (2018). Managing architectural technical debt: A unified model and systematic literature review. Journal of Systems and Software, 135, 1-16.

[13]. Martini, A., Bosch, J., & Chaudron, M. (2015). Investigating architectural technical debt accumulation and refactoring over time: A multiple-case study. Information and Software Technology, 67, 237-253.

[14]. Nord, R. L., Ozkaya, I., Kruchten, P., & Gonzalez-Rojas, M. (2012). In search of a metric for managing architectural technical debt. In Proceedings of the Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture (WICSA/ECSA) (pp. 91-100). IEEE.

[15]. Ernst, N. A., Bellomo, S., Ozkaya, I., Nord, R. L., & Gorton, I. (2015). Measure it? Manage it? Ignore it? Software practitioners and technical debt. In Proceedings of the 10th Joint Meeting on Foundations of Software Engineering (ESEC/FSE 2015) (pp. 50-60). ACM.

[16]. Seaman, C., & Guo, Y. (2011). Measuring and monitoring technical debt. Advances in Computers, 82, 25-46.

[17]. Seacord, R. C., Plakosh, D., & Lewis, G. A. (2003). Modernizing legacy systems: Software technologies, engineering processes, and business practices. Addison-Wesley.

[18]. Sneed, H. M. (1995). Planning the reengineering of legacy systems. IEEE Software, 12(1), 24-34.

[19]. Feathers, M. C. (2004). Working effectively with legacy code. Prentice Hall.

[20]. Fowler, M. (2018). Refactoring: Improving the design of existing code (2nd ed.). Addison-Wesley.

[21]. Brooks, F. P. (1987). No silver bullet: Essence and accidents of software engineering. Computer, 20(4), 10-19.

[22]. Walsh, J. P., & Ungson, G. R. (1991). Organizational memory. Academy of Management Review, 16(1), 57-91.

[23]. Nelson, R. R., & Winter, S. G. (1982). An evolutionary theory of economic change. Harvard University Press.

[24]. Basel Committee on Banking Supervision. (2013). Principles for effective risk data aggregation and risk reporting (BCBS 239). Bank for International Settlements.

[25]. Basel Committee on Banking Supervision. (2021). Principles for operational resilience. Bank for International Settlements.

[26]. H. Snyder, "Literature review as a research methodology: An overview and guidelines," Journal of Business Research, vol. 104, pp. 333–339, 2019, doi: 10.1016/j.jbusres.2019.07.039.

[27]. T. C. Lethbridge, J. Singer, and A. Forward, "How software engineers use

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 10 October 2026
Page No: 5796 - 5804
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0752

documentation: The state of the practice," IEEE Software, vol. 20, no. 6, pp. 35–39, Nov.–Dec. 2003, doi: 10.1109/MS.2003.1241364.

[28]. Runeson, P., & Host, M. (2009). Guidelines for conducting and reporting case study research in software engineering. Empirical Software Engineering, 14(2), 131-164.