engineering proceedings

MDPI

Proceeding Paper

Integrating the Approach of Adjustable Reliability into the System Development Life Cycle †

Edita Djambazova

Institute of Information and Communication Technologies, Bulgarian Academy of Sciences, 1113 Sofia, Bulgaria; edita.djambazova@iict.bas.bg

† Presented at the 15th International Scientific Conference TechSys 2026—Engineering, Technologies and Systems, Plovdiv, Bulgaria, 14–16 May 2026.

Abstract

Fault-tolerant distributed systems are implemented in safety-critical applications where a system failure could cause severe damage and threaten human lives. To guarantee their flawless operation, their dependability attributes must be embedded early and continuously throughout system design, rather than treating them as an afterthought. This paper presents a conceptual framework for integrating the approach of adjustable reliability into the System Development Life Cycle (SDLC). The approach of adjustable reliability provides a way to distribute structural hardware redundancy and achieve the system reliability required by the application. The proposed framework shifts reliability from a design add-on to a core architectural decision variable in the design of dependable distributed systems. Opportunities and challenges involved are discussed, and some future research directions are outlined.

Keywords: approach of adjustable reliability; dependable distributed systems; System Development Life Cycle (SDLC); conceptual framework; system design

1. Introduction

Dependable systems are intended to deliver a service that avoids more frequent or severe failures than is acceptable [1]. To achieve their high standards of reliability, they must be designed with dependability characteristics in mind. Attributes, such as reliability, availability, safety, etc., are part of system design requirements. Distributed systems for safety-critical applications must be fault-tolerant, and fault tolerance is embedded in their design from the very beginning. To achieve fault-tolerant component behavior in dependable distributed systems, they are constructed using replicated modules [2–6].

Introducing redundancy in computer systems as a fault-tolerance method and replication as its technical realization is well-known and thoroughly explored [5–12]. Although redundancy management has been studied and broadly implemented for many years, the design of new dependable distributed systems, the development of systems of systems, and cyber–physical systems [13] imply a new view and a search for new approaches to redundancy implementation. One such approach is to adjust system reliability by managing hardware redundancy according to application requirements [14].

This paper presents a conceptual framework to integrate the approach of adjustable reliability into the System Development Life Cycle (SDLC). The proposed framework shifts reliability from a design add-on to a core architectural decision variable and gives system designers opportunities to manage complex dependability specifications.

Academic Editors: Nikola G. Shakev, Valyo N. Nikolov, Nikolay R. Kakanakov and Sevil A. Ahmed-Shieva

Published: 22 July 2026

The paper is organized into five sections. Section 2 describes the redundancy management approaches. Section 3 presents the approach of adjustable reliability and introduces the conceptual framework for integrating it into the SDLC. A brief discussion of related work is provided in Section 4. The opportunities and challenges of the conceptual framework are discussed in Section 5. The paper concludes with Section 6.

2. Redundancy Management in Dependable Distributed Systems

Redundancy is a functionality or component of a computer system that adds resources for the delivery of its correct service. It is a method for implementing fault tolerance in dependable computer systems and is, in turn, realized by replication techniques [5,7,8,11,12]. This paper considers only hardware structural redundancy. It is implemented using different replication styles and degrees.

The replication style defines the way the replicated components perform their operation. The fault-tolerant components have replicated modules and only one of them, called primary, issues the output result. The other replicates are secondary. Depending on the replication style, the redundancy can be passive or active. Sparing is a passive form of redundancy [5,7,9]. Replication is a concurrent operation of identical modules that execute the same functions on the same input data and compare their results [5,7,9,11,12].

The replication degree determines the number of modules in a component. Depending on the importance of a component for the system operation, it can have one or more replicates or not be replicated at all. The replication degree depends on the fault-tolerance requirements as well.

In hardware, the replication takes the form of N-modular redundancy (NMR) [5,8,9,11,12]. It is mostly applied as dual and triple modular redundancy. In dual modular redundancy (DMR), the two replicas compare their results and, in case of discrepancy, the component does not issue any result, remaining fail-silent. In triple modular redundancy (TMR), there are three active modules and a voter. The active modules operate simultaneously, and the voter determines the majority result, which is issued to the object under control.

Dependable distributed systems can use equal [2,15–19] or different [14,16,18,20–24] redundancy degrees for all components. The approach of adjustable reliability is based on the notion of adjustability. Adjustability is the property of a dependable distributed real-time system to distribute the structural redundancy according to the reliability requirements of the application [14].

3. The Conceptual Framework

The necessity of having components with different replication degrees is related to their criticality. Unlike the approaches implementing mixed criticality to software application parts executed over evenly replicated hardware, the reliability adjustment approach proposes a component's redundancy degree to be determined according to its criticality at the design stage, and the fault-containment components to operate with mixed redundancy. A quantitative assessment of the implementation opportunities for dependable distributed systems with the approach of adjustable reliability is given in [14], and these systems are compared to systems without structural redundancy distribution. The comparison is conducted through simulation modeling, and the results show that there are redundancy distributions that achieve the total system reliability required by the application.

The fault-tolerant distributed system with adjustable reliability is built out of components whose fault tolerance is guaranteed by replication and self-checking. The components can have different degrees of replication depending on their criticality. By changing component structural redundancy at the design stage, a change of the system's reliability

according to the requirements of the application is achieved. Hardware faults are modeled, and the goal is to achieve hardware reliability.

3.1. The Approach of Adjustable Reliability

The fault-tolerant distributed system with adjustable reliability [14,25] applies different replication degrees to the hardware components that are built out of identical modules. A module is the smallest replaceable unit in the system. This notion is related to the replication techniques and the method of introducing redundancy in the system.

The approach of adjustable reliability is a way to determine the structural redundancy distributions that satisfy the requirement of system reliability of the application. The aim is to find the system configuration that achieves the required reliability—R_{total}.

The approach of adjustable reliability can be described with the following steps:

1. Determining R_{total}.
2. Determining the parameters describing the environment—fault types, fault rates.
3. Determining the system requirements—mean time to failure, mean time to repair, availability, mean downtime, possibilities for local and system repair, and the respective repair rates.
4. Determining the important control parameters and the critical components.
5. Determining the criticality levels.
6. Determining the system configurations with adjustable reliability that is equal to or greater than the total reliability R_{total}.
7. Checking which of the possible configurations fits in the best way into the application requirements.
8. In case no appropriate configuration is found, the input specifications are changed, and the procedure is repeated.

If more of the possible system configurations with adjustable reliability fulfill the requirement of system reliability under the identified specifications, the appropriate configuration for the application can be chosen according to additional criteria, such as the number of single, DMR, or TMR components, desired criticality levels, the highest total reliability, the longest operational period with high total reliability, etc. If none of the configurations of the system with adjustable reliability satisfy the requirement of R_{total} of the application, it is necessary to reconsider the system specifications, including the requirement of total reliability.

3.2. Conceptual Framework of Integrating the Approach of Adjustable Reliability into SDLC

The System Development Life Cycle [26,27] is a structured framework used to plan, design, build, test, and deploy information systems efficiently and effectively. It defines a series of phases that guide the development process from initial concept to system retirement, ensuring quality, reliability, and alignment with user requirements. Usually, the phases of SDLC include planning, analysis, design, implementation (or development), testing, deployment, and maintenance. In the planning phase, the project scope, objectives, feasibility, and resource requirements are defined. The analysis includes identifying user requirements and system specifications. During the design phase, the system architecture, data models, and detailed specifications for implementation are developed. The implementation includes building and integrating the system components according to the design. Testing is the process of verifying that the system meets requirements, is reliable, and functions correctly. System deployment is the release to users and performance of training, installation, and data migration. Maintenance includes monitoring system performance, fixing issues, and updating the system to adapt to changing needs.

https://doi.org/10.3390/engproc2026150059

SDLC provides a repeatable and organized approach to reduce errors, control costs, and ensure that the delivered system meets functional and quality expectations. It can follow different models, such as Waterfall, Agile, Spiral, or V-Model, depending on project complexity and flexibility requirements [28].

The approach of adjustable reliability can be incorporated into the SDLC, specifically in the first four phases, from idea to implementation. The conceptual framework of this integration is illustrated in Figure 1. The blue arrow in the figure shows the progression of the system development lifecycle. Its phases are also depicted in blue. The orange arrow and blocks in Figure 1 are related to the approach of adjustable reliability. The development steps and dependability characteristics defined in the respective phases are shown in gray.

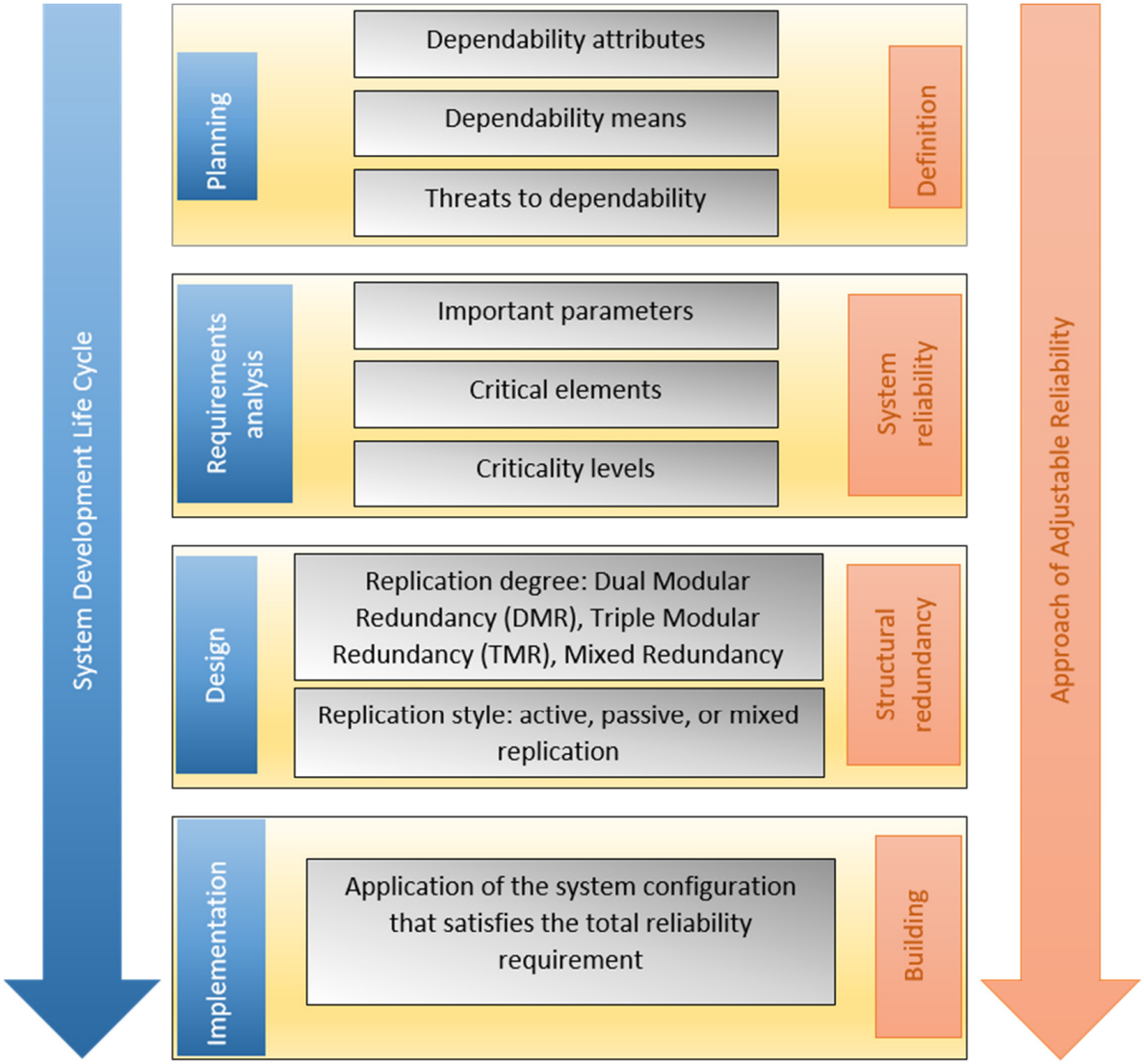

Figure 1. Conceptual framework for integrating the approach of adjustable reliability into the system development lifecycle.

The definition of dependability attributes, means, and threats to dependability relate to the planning phase of SDLC. Dependability attributes are reliability, availability, safety, integrity, and maintainability [1,5]. User requirements must outline their importance, targeted values, and prioritization. At this stage, the system reliability R_{total} should be determined. During planning, fault-tolerance methods, such as replication, error detection, error recovery, etc., are identified, since they impact the architectural solutions. In fault-

tolerant systems, dependability measures are directed to specific threats (faults, errors, and failures). Checking and control mechanisms depend on the fault types they must detect.

During the requirements analysis phase, all characteristics related to system reliability are defined. The important system parameters that must be monitored and controlled are determined here. Dependable distributed systems control safety-critical applications, e.g., industrial processes, autonomous vehicles, and critical infrastructures, and there are specific parameters to be kept within predefined bounds, or some thresholds must be respected. Likewise, critical system components must be identified to plan their protection. Criticality levels are related to fault rates and component importance. If a component controls an important parameter and/or is subjected to high fault rates, it must be made more fault-tolerant.

Decisions about the structural redundancy distribution of hardware components are made in the design phase. Here, the replication degree and style are determined. Components that control vital parameters can be built with DMR or TMR to ensure high reliability. Paired with self-checking tools, the constituent modules can provide additional fault coverage. If time constraints are stringent, active replication can be applied. If system operation and environmental conditions allow, some components can function with passive replication or no replication at all. In the early stages of system design, system configurations with possible hardware redundancy distributions must be modelled to determine which ones satisfy the application's reliability requirements. Analytical, simulation, and hybrid models can be applied to study system behavior and outline the appropriate structural redundancy configurations.

The selected system configuration that satisfies the total reliability requirement is built during the implementation phase of the SDLC. Here, the system architecture, with the determined dependability characteristics, becomes an engineering design compliant with the technical standards.

4. Related Work

Recent research has explored various approaches to improving fault tolerance and reliability in distributed and adaptive computer systems. Tang et al. [29] propose a distributed adaptive multipath redundant transmission mechanism aimed at increasing communication reliability by dynamically selecting multiple transmission paths and adjusting redundancy according to network conditions. In the context of embedded and multicore systems, Siyadatzadeh et al. [30] introduce RL-TIME, a reinforcement learning-based framework that dynamically determines task replication strategies to balance reliability, timing constraints, and resource utilization. Similarly, Murimi [31] investigates machine learning-driven replication strategies that leverage system monitoring data to adapt redundancy levels and improve fault resilience in large-scale distributed systems. Fault tolerance in AI-based systems is addressed by Rajagede et al. [32], who propose NAPER, a lightweight protection framework designed to safeguard deep neural network inference against hardware faults in resource-constrained real-time environments. In addition to these specific techniques, broader perspectives are provided by survey studies. De Souza and Ferrari [33] present a systematic review of fault tolerance techniques for adaptive and context-aware systems, identifying key categories such as replication, recovery mechanisms, reconfiguration, and graceful degradation, as well as common fault types in dynamic environments. Complementing technical approaches, Aguilar [34] explores process-oriented improvements for system reliability by applying Lean Six Sigma methodologies to reduce Mean Time to Recovery during system outages.

https://doi.org/10.3390/engproc2026150059

These studies demonstrate the growing importance of adaptive, data-driven, and multidisciplinary approaches for improving the dependability of modern distributed and cyber–physical systems. The use of intelligent approaches can help adapt system operation to meet the desired reliability and safety parameters. The runtime adjustment is usually implemented in software and requires the replica determinism to be guaranteed.

Integrating the approach of adjustable reliability into the SDLC allows hardware replication to be decided and verified in advance. The achieved flexibility is not comparable to software solutions, but it preserves predictable and deterministic system behavior.

5. Discussion

Integrating the approach of adjustable reliability into the system development lifecycle introduces flexibility, efficient resource allocation, and supports design decision-making in the early stages of the SDLC. It also poses some challenges, which are discussed in the next subsections.

5.1. Advantages of the Conceptual Framework

The presented conceptual framework has the advantages of enhanced flexibility, better resource efficiency, and improved dependability modeling. It allows the system to adapt reliability levels for different operational contexts. The system reliability requirement is addressed early in system design. Thus, the hardware structural redundancy configurations can be modeled, and important parameters and criticality levels can be determined. The simulation results in [14] outlined that no clear dependence between the structural redundancy distribution and the reliability can be drawn. The redundancy distribution impacts the system's reliability, and some distributions are more favorable for the system than others are. The choice of system configuration depending on application requirements can lead to system reliability and availability improvement.

The approach of adjustable reliability allocates redundancy where most needed, reducing unnecessary costs. Since this allocation is based on analytical and simulation results, its efficiency can be justified and improved.

Using simulation and modeling early in design supports informed trade-offs. Analytical models (e.g., reliability block diagrams, fault trees, simulation models) can be integrated to evaluate alternative reliability configurations before implementation. Simulation tools can be used to examine how changes in redundancy or component configurations affect system dependability and meet the adjustable reliability goals defined earlier.

Main contributions of the approach of adjustable reliability to system development are summarized in Table 1.

Table 1. Contributions of integrating the approach of adjustable reliability into the SDLC.

SDLC Phase	Contribution of Adjustable Reliability
Planning	Sets configurable dependability goals
Requirements	Identifies criticality levels
Design	Manages hardware structural redundancy
Implementation	Reliability embedded into architecture

5.2. Risks and Challenges

Integrating adjustable reliability into the SDLC provides flexibility and resource optimization, but it also introduces architectural, managerial, and verification challenges.

As with any dependable distributed system, the system with adjustable reliability applies redundancy. Introducing configurable redundancy degrees increases architectural complexity. It may cause design errors due to additional states and transitions. Adjustable

reliability multiplies the number of possible system states, which may render reliability modeling (e.g., Markov-based models) harder to scale and analyze. If simulation modeling is applied, simulation time and computational cost may increase.

When a reliability configuration is chosen, it must be verified. Testing effort grows exponentially with the number of tunable parameters (e.g., fault coverage and criticality level).

Integrating the approach of adjustable reliability into the SDLC may increase initial development cost, since additional modeling, simulation, and validation effort is needed. This may also make design cycles longer.

Adjustable reliability introduces multi-objective optimization. Reliability and resource usage must be efficient, and costs must be lowered. This may lead to performance degradation. A balance and possibly predictive models are needed to find the right design decision.

Table 2 summarizes the key challenges.

Table 2. Key challenges of integrating the approach of adjustable reliability into the SDLC.

Category	Main Risk
Architecture	Increased structural complexity
Verification	Combinatorial explosion of configurations
Performance	Trade-off management complexity
Economics	Higher development and maintenance costs

The challenges can be overcome with mitigation strategies. Increased structural complexity is a risk with dependable distributed systems. The shift-left strategy can help reduce the problem by executing simulation modeling earlier in design. Partitioning the system model into independent sub-models can reduce complexity and simulation time.

The number of tunable parameters can be limited depending on the application requirements. Analytical models and simulations can be combined to reduce computational burden. Another possible mitigation is to validate only extreme cases.

6. Conclusions

The introduced conceptual framework of integrating the approach of adjustable reliability into the system development lifecycle provides a flexible, resource-effective, and criticality-aware design strategy to develop dependable distributed systems. The framework regards only hardware structural redundancy, but it can be extended to software replication, where different assumptions apply. This can lead to increased flexibility and adaptability to runtime conditions.

The approach of adjustable reliability can be combined with intelligent technologies, such as artificial intelligence, to reduce design complexity and simulation time.

Funding: This research was funded by the Bulgarian National Science Fund (Grant agreement No. KP-06-H86/9 IS-PGR-SADOVO, project "Intelligent system for managing Bulgarian plant gene pool, conserved in the Genbank of IPGR-Sadovo", and the project "ReSearcH on formAl models for the oPtimization and pErsonalization of modern technological methods of STEM education (SHAPES)", contract КП-06-Н75/11/8 December 2023).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: The author declares no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

DMR	Dual Modular Redundancy
TMR	Triple Modular Redundancy
SDLC	System Development Life Cycle
NMR	N-modular redundancy

References

1. Avižienis, A.; Laprie, J.-C.; Randell, B.; Landwehr, C. Basic concepts and taxonomy of dependable and secure computing. *IEEE Trans. Dependable Secur. Comput.* **2004**, *1*, 11–33. [CrossRef]
2. Kopetz, H. *Real-Time Systems: Design Principles for Distributed Embedded Applications*, 2nd ed.; Springer: New York, NY, USA, 2011.
3. Barranco, M.; Derasevic, S.; Proenza, J. An architecture for highly reliable fault-tolerant adaptive distributed embedded systems. *Computer* **2020**, *53*, 38–46. [CrossRef]
4. Birman, K.P. *Reliable Distributed Systems: Technologies, Web Services, and Applications*; Springer: New York, NY, USA, 2005.
5. Dubrova, E. *Fault-Tolerant Design*; Springer: New York, NY, USA, 2013.
6. Erciyes, K. *Distributed Real-Time Systems: Theory and Practice*; Springer: Cham, Switzerland, 2019.
7. Geffroy, J.-C.; Motet, G. *Design of Dependable Computing Systems*; Springer: Dordrecht, The Netherlands, 2002.
8. Koren, I.; Krishna, C.M. *Fault-Tolerant Systems*, 2nd ed.; Morgan Kaufmann: Amsterdam, The Netherlands, 2021.
9. Lee, P.A.; Anderson, T. *Fault Tolerance: Principles and Practice*, 2nd ed.; Springer: Berlin, Germany, 1990.
10. Pradhan, D.K. *Fault-Tolerant Computer System Design*; Prentice-Hall: Upper Saddle River, NJ, USA, 1996.
11. Shooman, M.L. *Reliability of Computer Systems and Networks: Fault Tolerance, Analysis, and Design*; Wiley: Hoboken, NJ, USA, 2002.
12. Sorin, D.J. *Fault Tolerant Computer Architecture*; Morgan & Claypool: San Rafael, CA, USA, 2009.
13. Kopetz, H. *Simplicity Is Complex: Foundations of Cyber-Physical System Design*; Springer: Cham, Switzerland, 2019.
14. Djambazova, E. Achieving system reliability using reliability adjustment. In *Proceedings of the 23rd International Conference on Computer Systems and Technologies (CompSysTech'22), Ruse, Bulgaria*; ACM: New York, NY, USA, 2022.
15. Kopetz, H.; Damm, A.; Koza, C.; Mulazzani, M.; Schwabl, W.; Senft, C.; Zailinger, R. Distributed fault-tolerant real-time systems: The MARS approach. *IEEE Micro* **1989**, *9*, 25–40. [CrossRef]
16. Veríssimo, P.; Casimiro, A.; Pinho, L.M.; Vasques, F.; Rodrigues, L.; Tovar, E. Distributed computer-controlled systems: The DEAR-COTS approach. *IFAC Proc. Vol.* **2000**, *33*, 113–120. [CrossRef]
17. Randell, B.; Laprie, J.-C.; Kopetz, H.; Littlewood, B. *Predictably Dependable Computing Systems*; Springer: Berlin, Germany, 1995.
18. Herzner, W.; Schlick, R.; Schlager, M.; Leiner, B.; Huber, B.; Balogh, A.; Csertan, G.; LeGuennec, A.; LeSergent, T.; Suri, N.; et al. Model-based development of distributed embedded real-time systems with the DECOS tool-chain. *SAE Tech. Pap.* **2007**. [CrossRef]
19. Kopetz, H.; Bauer, G. The time-triggered architecture. *Proc. IEEE* **2003**, *91*, 112–126. [CrossRef]
20. Powell, D.; Arlat, J.; Beus-Dukic, L.; Bondavalli, A.; Coppola, P.; Fantechi, A.; Jenn, E.; Rabejac, C.; Wellings, A. GUARDS: A generic upgradable architecture for real-time dependable systems. *IEEE Trans. Parallel Distrib. Syst.* **1999**, *10*, 580–599. [CrossRef]
21. Pinho, L.M.; Vasques, F.; Wellings, A. Replication management in reliable real-time systems. *Real-Time Syst.* **2004**, *26*, 261–296. [CrossRef]
22. Obermaisser, R.; Peti, P.; Huber, B.; El Salloum, C. DECOS: An integrated time-triggered architecture. *Elektrotech. Inftech.* **2006**, *123*, 83–95. [CrossRef]
23. Dumitras, T.; Srivastava, D.; Narasimhan, P. Architecting and implementing versatile dependability. In *Architecting Dependable Systems III; LNCS 3549*; Springer: Berlin, Germany, 2005; pp. 234–254.
24. Narasimhan, P.; Dumitras, T.; Paulos, A.; Pertet, S.; Reverte, C.; Slember, J.; Srivastava, D. MEAD: Support for real-time fault-tolerant CORBA. *Concurr. Comput. Pract. Exp.* **2005**, *17*, 1527–1545. [CrossRef]
25. Djambazova, E. A fault-tolerant real-time system with adjustable reliability. In *Proceedings of the 22rd International Conference on Computer Systems and Technologies (CompSysTech'21), Ruse, Bulgaria*; ACM: New York, NY, USA, 2021.
26. Lisda, H.S.M.; Bayunanda, E.; Madhika, Y.R. Systematic literature review SDLC in software engineering. *Int. J. Comput. Inf. Technol.* **2023**, *12*, 31–42. [CrossRef]
27. Acharya, B.; Sahu, P.K. Software development life cycle models: A review paper. *Int. J. Adv. Res. Eng. Technol.* **2020**, *11*, 169–176.
28. Gorrod, M. *The Software Development Lifecycle, in Finance and Capital Markets Series*; Palgrave/Macmillan: London, UK, 2002.
29. Tang, Z.; You, J.; Li, Y. A distributed adaptive multipath redundant transmission mechanism for high-reliability communication. *Electronics* **2025**, *14*, 4735. [CrossRef]

30. Siyadatzadeh, R.; Ansari, M.; Shafique, M.; Ejlali, A. RL-TIME: Reinforcement learning-based task replication in multicore embedded systems. *arXiv* **2025**, arXiv:2503.12677. [CrossRef]
31. Murimi, A.K. Machine learning-driven replication strategies for enhancing fault tolerance in large-scale distributed systems. *arXiv* **2025**, arXiv:2511.11749.
32. Rajagede, R.A.; Santriaji, M.H.; Fikriansyah, M.A.; Nuha, H.H.; Fu, Y.; Solihin, Y. NAPER: Fault protection for real-time resource-constrained deep neural networks. *arXiv* **2025**, arXiv:2504.06591. [CrossRef]
33. De Souza, K.E.; Ferrari, F.C. A systematic review of fault tolerance techniques for adaptive and context-aware systems. In Proceedings of the 2022 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS), Virtual, 19–23 September 2022; pp. 21–30. [CrossRef]
34. Aguilar, A. Lowering mean time to recovery (MTTR) in responding to system downtime or outages: An application of Lean Six Sigma methodology. In *Proceedings of the International Conference on Industrial Engineering and Operations Management*; IEOM Society: Southfield, MI, USA, 2023; pp. 1234–1245.