
Digital Twin Based Framework for Real Time Software System Reliability Evaluation

Abstract

Modern software systems require continuous reliability evaluation because service failures can spread across connected components and interrupt operations. This study presents a digital twin framework for real-time software system reliability evaluation. The framework combines live monitoring data, synchronized virtual states, reliability scoring, and short-term behaviour analysis. Response time, error rate, resource utilization, service availability, failure events, and dependency health were used to represent system condition. The digital twin was evaluated during normal operation, increasing workload, service failure, and recovery, and was compared with a conventional threshold-based method. Results showed that the twin followed the physical system closely, detected reliability reduction earlier, responded faster to failure, and tracked recovery more accurately. The framework also provided a smoother reliability pattern by combining several operational signals instead of waiting for fixed warning limits. The proposed approach supports continuous monitoring, early warning, failure analysis, recovery tracking, and better operational decisions in complex software environments.

Keywords: Digital twin, Software reliability, Real-time monitoring, Failure detection, System recovery

1. Introduction

Modern software systems support banking, healthcare, transportation, communication, and many other important services. These systems often contain several connected services that operate across cloud, edge, and local computing environments. A failure in one service can increase response time, produce errors, or interrupt the complete application. Digital twin architectures can create a continuously updated virtual representation of a running system and support monitoring, analysis, and operational decisions [1]. This approach can help software teams understand reliability changes before they become serious service failures.

Traditional monitoring tools commonly compare individual measurements with fixed warning limits. These tools can show high CPU usage, slow response time, or increased error rates, but they may not explain how several changes are connected. A digital twin can combine live measurements with the structure and expected behaviour of the software system. Microservice and serverless technologies can support scalable digital twin components that process distributed events and respond to changing workloads [2]. A combined system view can therefore provide more useful reliability information than separate infrastructure alerts.

Real-time reliability evaluation requires continuous synchronization between the operating software and its virtual model. The twin must receive updated response times, error rates, resource conditions, availability records, and dependency states. The closeness of the digital twin to the original system can be measured through the consistency and timing of their exchanged information [3]. Poor synchronization may produce outdated reliability estimates and delay the detection of abnormal behaviour.

A software digital twin must also support simulation and prediction rather than simply copying monitoring data. Virtual experiments can test the likely effect of increasing workload, service interruption, resource shortage, or dependency failure without disturbing the live system. Digital twin prototypes can support automated testing and evaluation by reproducing important operating conditions in a controlled software environment [4]. This study proposes a digital twin framework that evaluates system reliability in real time and tracks changes during normal operation, increasing load, failure, and recovery.

2. Methodology

The proposed methodology contained software-system modelling, live-data collection, twin synchronization, reliability estimation, failure simulation, and real-time evaluation. The monitored application was represented as connected services, databases, communication interfaces, and external dependencies. Each software component was linked with a corresponding object in the digital twin. A

shared digital twin platform can manage reusable models, monitoring services, communication functions, and computational resources [5]. The proposed framework used this service-based structure to maintain the virtual representation throughout system operation.

Live measurements were collected from application performance monitoring tools, service logs, resource monitors, health-check interfaces, and distributed tracing systems. Response time represented the delay experienced by service requests, while error rate measured the proportion of failed operations. Resource utilization included processor, memory, network, and storage conditions. Availability was obtained from service health checks and successful request records. Failure and dependency information was collected from exception logs, service status messages, and communication traces.

The live data was transferred to the digital twin through an event-streaming layer. Every incoming event contained a component identifier, timestamp, metric type, and measured value. Events were arranged according to their time of occurrence, and delayed records were marked before processing. Digital twin entanglement describes how closely the virtual system follows the state and behaviour of the original system [6]. The synchronization layer therefore checked data delay, update frequency, and state agreement before calculating reliability.

Table 1 presents the five signal groups used for real-time evaluation. Response time and error rate described the direct behaviour observed by users and connected services. Resource utilization represented the operational load placed on the infrastructure, while availability showed whether the required services remained accessible. Failure events and dependency health were combined into one row because both described interruptions and risks that could spread across connected components. Each signal was converted into a normalized digital twin variable before being included in the reliability model.

Table 1. Digital Twin Reliability Signals, Live Monitoring Sources, Synchronization Method, and Reliability Contribution

Live signal	Monitoring source	Synchronization method	Digital twin variable	Reliability meaning	Evaluation contribution
Response time	Application performance monitor and distributed trace	Timestamp-aligned event update	Normalized service latency	Indicates service speed and user delay	Reduces reliability when latency exceeds the expected range
Error rate	Application logs and request records	Continuous error-count synchronization	Failed-request ratio	Represents incorrect or unsuccessful operations	Detects functional instability and service degradation
Resource utilization	CPU, memory, network, and storage monitors	Fixed-interval state update	Resource pressure index	Shows infrastructure stress	Identifies reliability risk caused by overload

Service availability	Health-check interface and uptime monitor	Real-time status synchronization	Availability probability	Represents continuous service access	Measures the ability to deliver the required function
Failure events and dependency health	Exception logs, service status, and communication traces	Event-triggered dependency update	Failure propagation index	Shows component failure and connected-service risk	Identifies local faults and their possible system-wide effect

The digital twin calculated one reliability score for every monitoring interval. The score combined normalized availability, error rate, response time, resource pressure, and failure propagation. Larger weights were assigned to service unavailability and confirmed failure events because they had a direct effect on system operation. Digital twin fault-monitoring methods can combine synchronized observations with virtual system behaviour to identify abnormal operating conditions [7]. The score was updated whenever new live information entered the synchronization layer.

The twin also simulated short-term operating behaviour using the latest system state. It estimated whether the current workload and dependency conditions were likely to produce further reliability reduction. Lightweight twin models can support continuous health monitoring when complete data transfer or heavy computation is not practical [8]. The proposed framework used recent metric windows and component relationships to generate short-horizon reliability forecasts.

The framework was evaluated during four operating conditions: normal service, increasing workload, controlled service failure, and recovery. The physical software measurements were compared with the digital twin reliability score and a conventional threshold-based monitoring method. Real-time digital twin frameworks can combine current and historical information to detect abnormal conditions and support fast operational decisions [9]. Evaluation measures included detection delay, reliability-score error, failure recognition rate, recovery tracking, and state-synchronization accuracy.

3. Results and Discussion

The proposed digital twin framework was evaluated across four operating stages: normal service, increasing workload, service failure, and system recovery. Reliability scores were calculated at successive monitoring intervals using response time, error rate, resource utilization, service availability, and dependency health. The digital twin score was compared with the reliability level observed in the physical software system and with a conventional threshold-based monitoring method. The results showed that the digital twin followed the real system condition more closely throughout the complete operating sequence.

During normal operation, all three methods reported high reliability because response time, error rate, and resource use remained within expected limits. As workload increased, the physical system reliability began to decline because service delay and resource pressure became higher. Figure 1 shows

that the digital twin detected this reduction earlier than the threshold-based method. Its score changed gradually with the real system, while the benchmark remained almost unchanged until fixed warning limits were crossed.

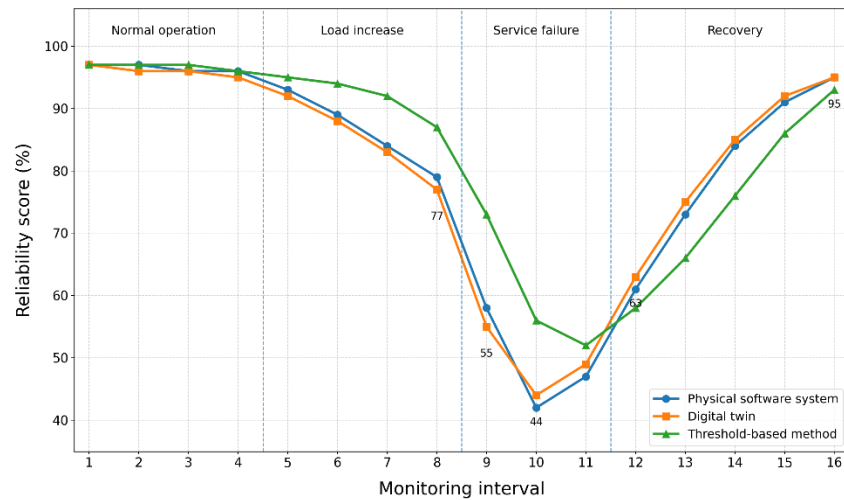


Figure 1. Real-Time Reliability Score During Normal Operation, Load Increase, Failure, and Recovery

The largest difference appeared when the service failure occurred. The physical system reliability dropped sharply, and the digital twin produced a similar decrease within the next monitoring interval. The threshold-based method responded later because it required several measurements to exceed their limits before lowering the reliability score. The digital twin also identified the effect of dependency failure and error propagation, which provided a more complete view of the system condition.

During recovery, the physical system gradually returned to normal as the failed service was restored and resource pressure decreased. The digital twin tracked this improvement closely and increased its reliability score as new healthy measurements became available. The threshold-based method recovered more slowly because it continued to reflect earlier warning states for several intervals. These results show that the digital twin can support faster reliability evaluation and more accurate tracking of changing software conditions.

4. Conclusion

This study presented a digital twin framework for real-time software system reliability evaluation. The framework combined live monitoring data, synchronized virtual system states, reliability scoring, and short-term behaviour analysis. It used response time, error rate, resource utilization, service availability, failure events, and dependency health to represent the current condition of the software system. The digital twin continuously updated its virtual state as new monitoring information became available. This

design allowed the framework to evaluate both local component behaviour and wider system-level reliability changes.

The results showed that the digital twin followed the physical software system more closely than the conventional threshold-based method during normal operation, workload increase, service failure, and recovery. It detected reliability reduction earlier, responded faster to failure, and tracked system recovery more accurately. The framework also provided a smoother and more realistic reliability pattern instead of waiting for fixed warning limits to be crossed. This feature can help operators recognize gradual degradation before it becomes a major service interruption. The proposed framework can therefore support continuous monitoring, early warning, failure analysis, recovery tracking, and better operational decisions in complex software systems.

References

1. Ferko, E., Bucaioni, A., & Behnam, M. (2022). Architecting digital twins. *Ieee Access*, 10, 50335-50350.
2. Bellavista, P., Bicocchi, N., Fogli, M., Giannelli, C., Mamei, M., & Picone, M. (2024). Exploiting microservices and serverless for digital twins in the cloud-to-edge continuum. *Future Generation Computer Systems*, 157, 275-287.
3. Bellavista, P., Bicocchi, N., Fogli, M., Giannelli, C., Mamei, M., & Picone, M. (2024). Ode: A metric for digital twin entanglement. *IEEE Open Journal of the Communications Society*, 5, 2377-2390.
4. Barbie, A., & Hasselbring, W. (2024). From digital twins to digital twin prototypes: Concepts, formalization, and applications. *IEEE access*, 12, 75337-75365.
5. Talasila, P., Gomes, C., Mikkelsen, P. H., Arboleda, S. G., Kamburjan, E., & Larsen, P. G. (2023, August). Digital twin as a service (DTaaS): a platform for digital twin developers and users. In *2023 IEEE Smart World Congress (SWC)* (pp. 1-8). IEEE.
6. Miao, Z., Li, W., & Pan, X. (2024). Multivariate time series collaborative compression for monitoring systems in securing cloud-based digital twin. *Journal of Cloud Computing*, 13(1), 16.
7. Bofill, J., Abisado, M., Villaverde, J., & Sampedro, G. A. (2023). Exploring digital twin-based fault monitoring: Challenges and opportunities. *Sensors*, 23(16), 7087.
8. Montana, F., Hartwell, A., Jacobs, W., Kadirkamanathan, V., Mills, A. R., & Clark, T. (2023). Through-life monitoring of resource-constrained systems and fleets. *arXiv preprint arXiv:2301.01017*.
9. Jameil, A. K., & Al-Raweshidy, H. (2025). A digital twin framework for real-time healthcare monitoring: leveraging AI and secure systems for enhanced patient outcomes. *Discover Internet of Things*, 5(1), 37.