Towards Measuring Digital Twins Fidelity at Runtime

Paula Muñoz
paulam@uma.es
ITIS Software, Universidad de Málaga
Spain

Javier Troya
jtroya@uma.es
ITIS Software, Universidad de Málaga
Spain

Antonio Vallecillo
av@uma.es
ITIS Software, Universidad de Málaga
Spain

Abstract

This paper introduces a novel approach for runtime validation and anomaly detection in Digital Twins. We enhance the trace alignment capabilities of the Needleman-Wunsch dynamic programming algorithm to enable continuous system state monitoring. Our method overcomes the limitations of previous works by eliminating the need for time series preprocessing or predefined behavioral constraints. By aligning traces and utilizing sliding windows, we periodically analyze the most recent snapshots to detect anomalies, delays, and deviations between the twins at runtime. This technique improves anomaly detection accuracy and system diagnostics by leveraging the behavioral duplication inherent in Digital Twins. We validated our prototype with elevator behavioral traces, demonstrating its effectiveness in measuring behavioral fidelity and monitoring system safety.

CCS Concepts

• **Software and its engineering → Operational analysis**.

Keywords

Digital twins, runtime monitoring, anomaly detection, validation.

ACM Reference Format:
Paula Muñoz, Javier Troya, and Antonio Vallecillo. 2024. Towards Measuring Digital Twins Fidelity at Runtime. In *ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems (MODELS Companion '24), September 22–27, 2024, Linz, Austria.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3652620.3688267

1 Introduction

Digital Twins have emerged as a trending practice for designing, maintaining, and operating complex systems in various domains, including manufacturing, energy, construction, and human health [2]. As a relatively new concept in the literature, many definitions have been proposed to describe them. This work uses the following terminology: a *Digital Twin* (DT) is a virtual representation of a real-world entity or process (the *Physical Twin*, PT), synchronized at a specified frequency and fidelity [3]. The twinned systems (DT and PT), the connections between them, and the set of system services comprise the so-called Digital Twin System (DTS) [10].

Depending on the system's purpose, the DT replicates various characteristics of its physical counterpart, such as appearance, constraints, and behavior [2]. This replication is the key potential of the DT paradigm, as it reduces costs and risks throughout the system's lifecycle. By exploring potential scenarios without endangering the actual system, the DTS can predict and prevent potentially unsafe states at runtime [4].

This potential is especially leveraged in complex systems such as Cyber-Physical Systems (CPSs), which interact with uncertain environments. Ensuring that both the DT and PT exhibit faithful behavior is crucial in such contexts. Otherwise, the system might provide unreliable predictions or recommendations. Various methods exist in the literature to validate CPS behavior, including anomaly detection [6], runtime monitoring using constraints [7], and Complex Event Processing (CEP) [5].

Although widely used in current practices, these methods have some limitations. Anomaly detection typically involves analyzing the system's history to identify non-compliant behavior, often requiring preprocessing the time series to remove trends [6]. Runtime monitoring and CEP necessitate defining thresholds, temporal constraints, or rules to specify the expected behavior. These approaches might not account for uncommon scenarios or corner cases, potentially leaving the system vulnerable [7]. Additionally, these methods do not leverage the potential existence of a virtual replica, which DTSs include. They primarily determine whether the current state of the system is valid based on predefined rules, constraints, or historical data, rather than utilizing continuous, dynamic feedback available from a DT.

There are a few studies in the literature addressing validation in the context of DTs [2]. For example, Xu et al. [14] propose a method that uses Generative Adversarial Networks to automatically generate a replica of the PT's behavior and detect anomalous readings. However, this method only detects anomalies and does not account for acceptable delays between the twins. Moreover, trace comparisons leave no room for small delays between the recorded states of the twins but require that they happen at exactly the same time. This is often unrealistic, especially since there may be small delays when the communication used to transmit the states is via TCP or similar mechanisms or when the clocks of the two twins are not perfectly synchronized. To partly address these issues, Lugaresi et al. [8] suggested variations of the dynamic programming algorithms Longest Common Subsequence (LCSS) and Dynamic Time Warping (DTW) to analyze the runtime behavior of the twins. This approach provides a similarity score from 0 to 1 but, unfortunately, does not allow identification of the regions that produce the dissimilarities. Then, in a previous work [11, 13], we introduced a trace alignment algorithm based on another dynamic programming algorithm, namely Needleman-Wunsch, to identify anomalies or delays that overcome these problems, but it is limited to offline analysis, i.e., it was not designed for runtime monitoring.

This paper addresses the mentioned shortcomings by leveraging the DTs' ability to duplicate the PT's behavior to identify deviations

MODELS Companion '24, September 22–27, 2024, Linz, Austria

ACM ISBN 979-8-4007-0622-6/24/09
https://doi.org/10.1145/3652620.3688267

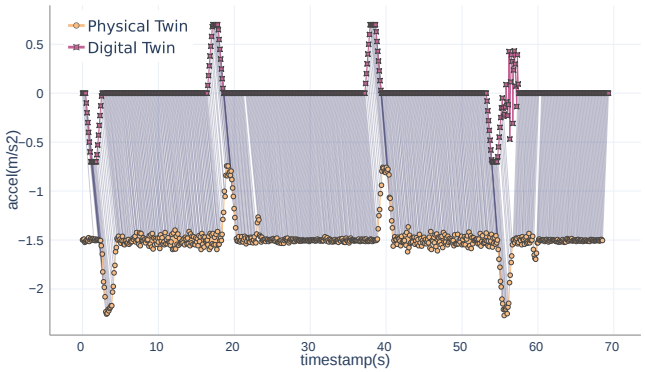

Figure 1: Trace alignment for Scenario (4-0-4).

between the twins at runtime, detect delays, and diagnose anomalies. Our approach does not require time-series preprocessing or the definition of behavioral constraints and rules. We enhance our previous proposal [11, 13], by performing alignments at runtime for continuous monitoring of the system's state, measuring behavioral fidelity between the twins, and using *sliding windows* to analyze the last n snapshots periodically.

This short paper presents the first application and prototypical implementation of this approach, which has been demonstrated using one exemplar system: an elevator.

2 Background

2.1 Running example: An Elevator

The elevator subject to study is located in a four-story building at the University of Mondragón in Spain, operating between floors 0 and 4. It serves students who cannot use the stairs and maintenance staff to transport heavy equipment. To ensure its reliability, the university wants to deploy a DTS to monitor the elevator's operation and maintenance, aiming to minimize downtime.

The virtual replica is the commercial simulator *Elevate*, which reproduces the acceleration between floors. This acceleration data helps determine the elevator's speed and position to derive equipment degradation and optimize its configuration.

Figure 1 shows the acceleration pattern for the elevator's journey from the 4th floor to the ground floor (timestamps 0.5 to 18.5 in DT), a stop, and the return to the 4th floor (timestamp 37.3 to 55.3 in DT). The acceleration peaks are divided into two groups: descent and ascent. The first peak indicates the descent's negative acceleration, and the second peak marks the positive acceleration required to brake on the ground floor. The latter two peaks correspond to the ascent back to the 4th floor.

Despite some noise in the PT's data, both twins' transition peaks are similar. However, the PT exhibits a unique behavior that the DT does not replicate. After braking in either direction, an additional small acceleration pattern is introduced to smooth the stop and enhance the user experience. This pattern, specific to this elevator model, should be noted as a discrepancy between the twins' traces. Figure 1 also shows a small initial delay in the PT behavior, due to the elevator start delay. Finally, we included a synthetic anomaly in the DT's trace from timestamp 55.4 to 57.4 to illustrate additional aspects of our approach.

2.2 Measuring Fidelity using Trace Alignment

2.2.1 Alignment Algorithm. Recently, we presented a proposal to measure the fidelity of the behavior of two twins using a trace alignment algorithm [13]. A trace is a sequence of consecutive snapshots, each representing the system's state when it was recorded. These snapshots are sampled with a constant period, generating a trace of equally spaced snapshots. In Figure 1, we present the traces of the PT and DT for the scenario (4-0-4). Each marker in the traces represents a snapshot containing the properties of interest that describe the system's state. For the elevator's example, as we are interested in its trajectory, we consider the timestamp and the acceleration.

Using this discrete behavior representation, we define a function to measure the similarity between each snapshot and determine their alignment based on a threshold known as the Maximum Acceptable Distance (MAD). The MAD is the maximum absolute difference between each of the properties of interest, typically 2 or 3 times the precision of the measurement tool used [13]. In the elevator case, the MAD is set to $0.15m/s^2$, given the accelerometer's precision of $0.05m/s^2$.

Once the similarity level between snapshots is calculated, we apply a trace alignment algorithm to find the optimal alignment between traces, maximizing the similarity between the pairs of aligned snapshots. The alignment algorithm is adapted from the dynamic programming algorithm Needleman-Wunsch (NDW) [12], originally used for the global alignment of sequences of characters. Our algorithm aligns each snapshot with at most one snapshot of the other trace or with a gap, depending on their similarity. If the snapshots are sufficiently similar, meaning their difference is smaller than the MAD, they are paired as a match. If their difference exceeds the MAD, they may be paired as a mismatch or remain unpaired, creating a gap.

In Figure 1, the alignment shows over 96.25% of matched snapshots, 0.72% of mismatched snapshots, and 3.02% gaps. Mismatches typically occur when adjacent snapshots match, but some middle snapshots of both traces do not fall within the MAD. These mismatches can be interpreted as anomalies. Since the penalty for gaps is higher than for mismatches, gaps are included only when no unpaired snapshots are available in the neighboring snapshots of the other trace. Gaps usually appear when there are delays in the behavior. For instance, Figure 1 shows that 3% of the gaps are distributed between the beginning of the alignment and the synthetic gap. We are recording the twins' behavior in the same period. At the beginning of the trace, the PT's descent starts a few seconds later, so corresponding snapshots of the DT's trace are missing, which are thus labeled as gaps. Similarly, a few extra snapshots appear at the end of the DT's trace to compensate for the initial delay. The anomalous snapshots are labeled as gaps since the corresponding simultaneous snapshots align with the extra ones. This could be interpreted as a stuttering situation that is corrected afterwards. The mismatches are located in the braking pattern of the elevator, pointing to an anomaly that does not exist in the other trace.

2.2.2 Fidelity Metrics. In our previous work [13], we measured the fidelity between the PT and DT using the *percentage of matched snapshots* (%MS) and a set of distance metrics to evaluate the similarity between aligned snapshots. The %MS indicates the number of snapshots within the Maximum Acceptable Distance (MAD), with

higher percentages representing closer trajectories. If the %MS is too low, it may be insufficient to evaluate the behavior accurately using distance metrics.

The distance measures we used are the *Frèchet* (FD) and the *Euclidean* (ED) distances [9]. The Frèchet distance calculates the maximum distance between all matched snapshots, while the Euclidean distance calculates the average distance between matched pairs. In an ideal alignment, the %MS would be 100%, and both FD and ED would be 0. Any decrease in %MS and increase in these distances indicates a lower level of fidelity. We previously determined that a system with more than 95% of matched snapshots represents a good alignment. If the %MS is between 90% and 95%, we then consider whether the FD and ED are within acceptable ranges for our application.

In the case of the alignment in Figure 1, the %MS is below 90% due to the initial delay, making the alignment unsatisfactory. However, if we exclude the snapshots representing the delay (2.3% of gaps), the %MS rises above 90% with an FD of 0.14 and an ED of 0.02, which are acceptable for our application.

3 Proposal

3.1 Description

We propose performing this analysis at runtime to enable continuous anomaly detection monitoring and correct the twins' behavior, avoiding potentially unsafe states. Our algorithm [13] has a time complexity of $O(n^2)$ because it uses dynamic programming to determine the optimal alignment by calculating the similarity of every pair of snapshots. Aligning the entire trace plus one snapshot every time a new one is sampled would lead to an increasingly slow response from the monitoring system.

Our solution performs these alignments using the n most recent snapshots, employing a concept known as *sliding windows*. Sliding windows are common in fields like event processing [5]. A *window* is a group of consecutive events defined by a time interval with a starting and ending time or by a number of events. Different types of sliding windows exist, depending on how the window's boundaries are defined. In this work, we assume that events are received at regular rates and that the sliding windows have a fixed number of snapshots, referred to as the *interval duration*, and are opened at regular time intervals, called the *interval period*.

In each interval, we use the NDW algorithm to calculate the alignment between snapshots included in the interval and apply two optimization techniques borrowed from the BLAST algorithm [1]: low-complexity regions and affine gaps (see [13] for a complete description of our algorithm). Once the two traces are aligned, we compute the fidelity metrics for the interval, namely, the number of matched snapshots (%MS) and the two distances, FD and ED. Finally, based on these values, we determine the potential existence of anomalies or delays.

3.2 Evaluation

To demonstrate and evaluate our proposal, we simulate that the traces of scenario (4-0-4) are received as marked in their snapshots. In the following subsections, we explore three different configurations for this scenario to showcase the effect of discrepancies, delays, and configuration parameters on the fidelity metrics.

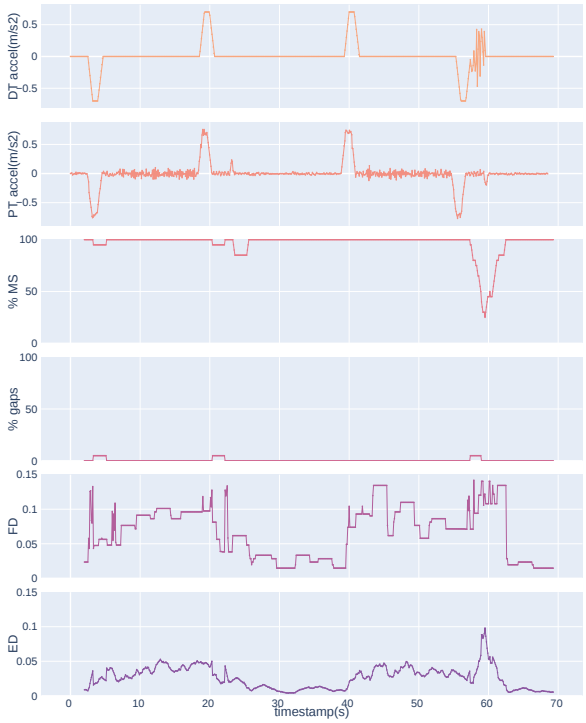

Figure 2: Metrics for Scenario (4-0-4) *without delay*, using sliding windows with an interval duration of 20 snapshots (2 seconds).

In the first analysis, we remove the 2-second delay from the trace to highlight the potential for anomaly detection without the delay's influence. The second analysis includes the delay, allowing us to compare the results. Both analyses consider an interval duration of 20, meaning we consider the last 2 seconds of snapshots in each alignment—the sampling period for the snapshots is 0.1 for both twins. In the final example, we increase the interval duration to 100 snapshots to see how the length of the intervals affects the analysis. In all cases, the interval period is set to 1, ensuring we analyze the trace thoroughly without missing any snapshots. If the period was greater than the duration, some snapshots might not be included in the alignments.

3.2.1 Case 1: No delay, interval duration: 20 snapshots. Let us begin aligning the traces without including the 2-second delay. The idea is to demonstrate our approach with perfectly synchronized traces of the system that include the small glitches and the final anomaly but without gaps. We performed the simulated runtime analysis using the same MAD value as the alignment described in Section 2.2.1. The results are shown in Figure 2. Some of the acceleration curves show a drop of around 5% in %MS, corresponding to 1 snapshot in each window. Figure 3 shows the aligned snapshots for three sliding windows. The first window covers a 2-second period from timestamps 2.5 to 4.5 and contains one unpaired snapshot for each trace: the third snapshot of the DT and the last snapshot of the PT. These gaps are due to some variability in the sampling process.

In the case of the two anomalous glitches (the small acceleration patterns implemented by the physical elevator to enhance the user experience) and the final synthetic anomaly that we introduced, they produce a drop in the percentage of matches, which is

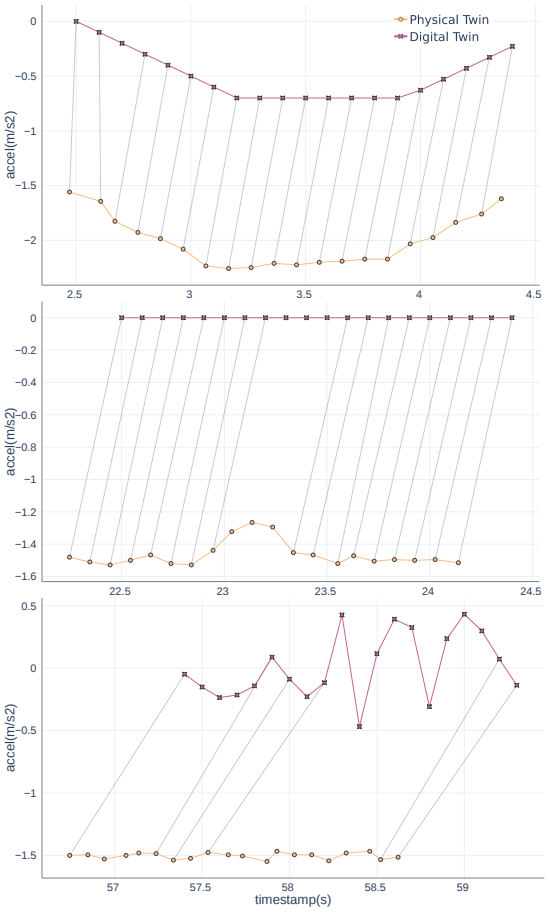

Figure 3: Alignments for Scenario (4-0-4) *without delay*, using sliding windows with interval duration of 20 snapshots (2 seconds).

translated into an increase in the percentage of mismatches and no increase in gaps. This leads us to classify them as anomalies. The first two (the small glitches) produce small variations in the fidelity metrics, while the third one (the big anomaly introduced synthetically) produces a significant decrease in the metrics, indicating a clear anomaly. This is showcased in the second and third windows of Figure 3. In the second window, which covers the period from timestamps 22.5 to 24.5, three mismatches indicate that the small acceleration pattern is an anomaly. Moving on to the third window, which spans from seconds 57.4 to 59.4, we observe that most of the synthetic noise does not match. The synthetic noise comprises random values within the working range, so only the values closer to 0 within the MAD range are aligned.

3.2.2 Case 2: Initial 2-sec delay, interval duration: 20 snapshots. Now, we analyze the effect of a 2-second delay in the PT trace with respect to the PT. The results, shown in Figure 4, indicate that the %MS drastically drops during each acceleration transition. The interval duration is too short to account for the 2-second delay in the PT's trace. This issue is illustrated in Figure 5, showcasing some window alignments.

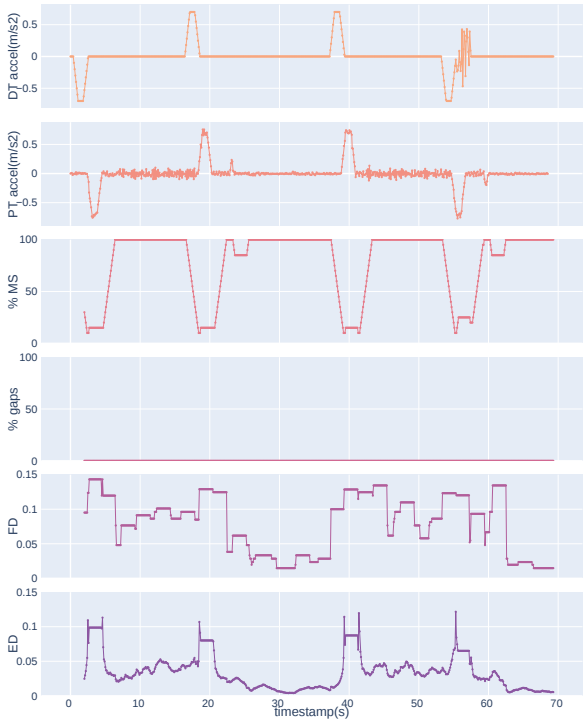

Figure 4: Metrics of Scenario (4-0-4) using sliding windows with an interval duration of 20 snapshots (2 seconds).

In the first window, covering the first two seconds, we see the initial acceleration transition in the DT, but only stationary behavior in the PT. Similarly, in the second window, from approximately 2.5 to 4.5 seconds, the same transition appears in the PT trace, without overlapping any part of the DT transition. This mismatch happens in all acceleration transitions, leading to significant drops in the %MS. In this analysis, the algorithm aligns only zero-acceleration areas, as seen in the third window from 4 to 5.5 seconds.

The drop in %MS is less severe during the smaller brake acceleration patterns. For example, the first brake pattern occurs around 23 to 23.5 seconds. One of the windows containing this pattern is the fourth shown in Figure 5. As an anomaly, this is detected as a mismatch, but the drop is smaller because the trace represents a smaller portion of the window, unlike the other acceleration patterns, which constitute around 80% of the window, resulting in a more drastic drop.

In this configuration, when we look at the distance metrics (FD and ED), we see that the highest values for both metrics happen at the same time as %MS significantly drops during the acceleration transitions. This occurs because in those sections, as shown in the third window of Figure 5, only a few points of stationary behavior are aligned, which fall within the range [0, 0.05] and result in the lowest distance values.

3.2.3 Case 3: initial 2-sec delay, interval duration: 100 snapshots. As we increase the interval duration to 100 snapshots, surpassing the delay between the windows, the results in Figure 6 show no drastic drops in %MS, which does not go below 77%. In such cases, the remaining 20% of snapshots are gaps, equating to 20 snapshots or approximately 2 seconds, corresponding to the delay. This is more clearly illustrated in the windows shown in Figure 7.

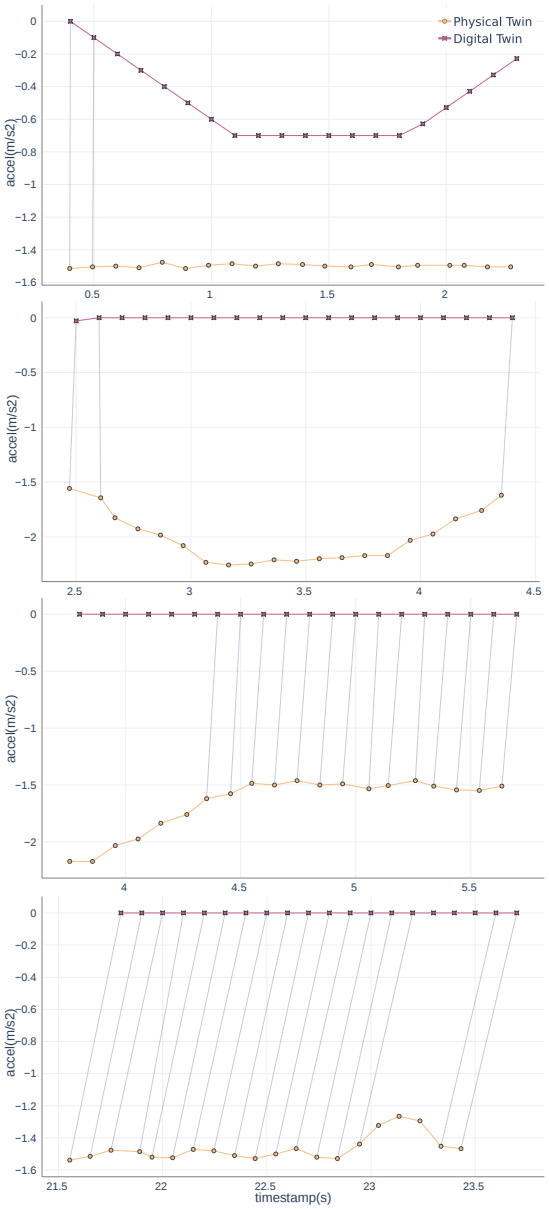

Figure 5: Alignments for Scenario (4-0-4) using sliding windows with an interval duration of 20 snapshots (2 seconds).

In the first window, covering the first 10 seconds of both traces, the transitions are aligned, and two sections of each trace are considered gaps. These gaps are precisely the initial snapshots of the PT's trace, which are absent in the DT due to the delay. In the second window, progressing 2 seconds in time, the acceleration transition of the DT is left behind, and only the PT's transition is included, now considered a mismatch. A similar situation occurs in the third window, at the next DT transition around 16 seconds. These progressive movements of the window produce the value transitions in the %MS seen in Figure 6. Finally, in the fourth window, between seconds 14 and 24, the first braking anomaly is detected as a set of mismatches. This window also shows the 20% of gaps, indicating the consistent 2-second delay throughout the trace. Increasing the

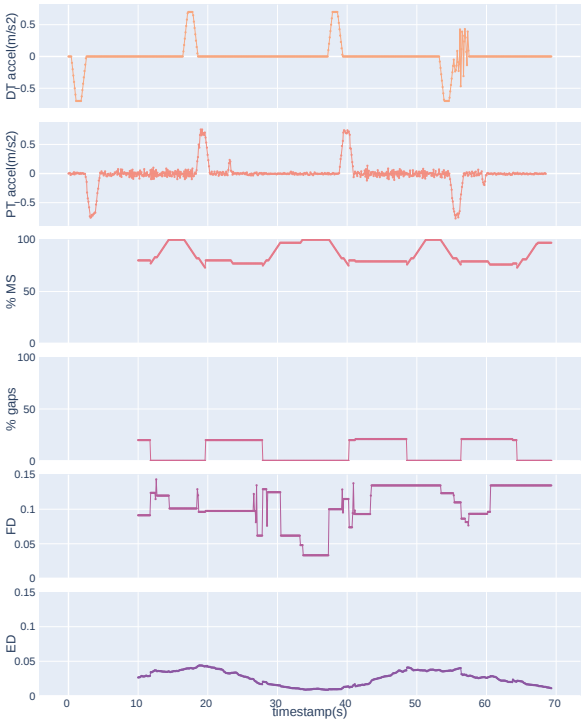

Figure 6: Metrics for Scenario (4-0-4) using sliding windows with an interval duration of 100 snapshots (10 seconds).

window allows us to accept a certain delay, enabling the alignment of common parts, such as acceleration curves, even if they are not perfectly synchronized. This results in a consistent number of gaps in certain areas, indicating the existence of a delay.

In Figure 6, the FD remains in the same range as in the previous case, with similar values, while the ED is smoothed, fluctuating only between 0 and 0.05. As the number of points in the average increases, its value smooths out. The longer the interval duration, the less significant the drops in metrics become, as it smooths out the average values of the distance metrics and the %MS. A longer window masks the impact of a few mismatches, gaps, or pairs of dissimilar pairs. We should keep this in mind when setting alarm thresholds and determining the interval duration for the detection of anomalies or delays.

4 Conclusions and future work

In this paper, we have introduced a new runtime validation approach for DTs. This method compares the PT and DT's behavioral traces to measure their similarity and diagnose anomalies, delays, and discrepancies. We present our first proof of concept using a real-world elevator operation scenario, where the algorithm successfully identified various anomalies and a delay. We have demonstrated the importance of aligning the traces when using sliding windows in the context of digital twins, because otherwise any delay or lack of synchronization, which is rather common in these environments, can invalidate the monitoring. This issue is not considered in most existing proposals. In addition, our approach uses various fidelity metrics to identify and quantify potential anomalies. This is essential to decide the degree of severity of the anomalies and raise the corresponding alarms or mitigating effects when necessary.

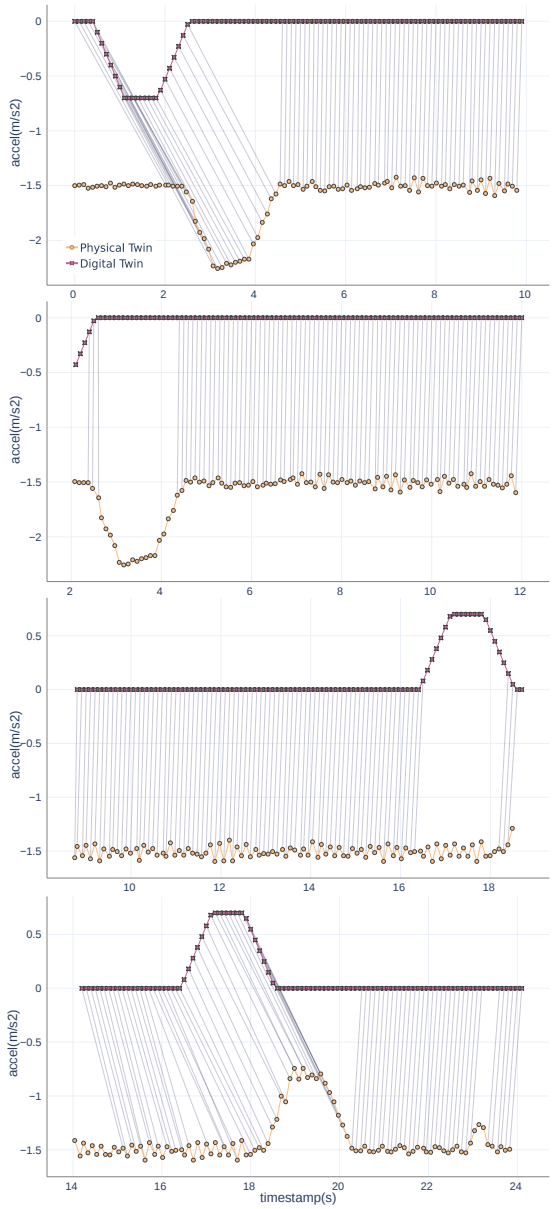

Figure 7: Alignments for Scenario (4-0-4) using sliding windows with an interval duration of 100 snapshots (10 seconds).

The example presented in this paper has served to illustrate our initial proposal, demonstrate its applicability, and show its advantages. However, we now need to conduct further research. For example, we need to study how to determine the appropriate values for the interval duration, depending on the specific system, its environment, and the properties of interest. We also need to investigate how to set the alignment algorithm parameters [13] in this context to deal with delays, e.g., depending on how strict we are about the twins' synchronization. We must also define guidelines for setting thresholds to raise alarms based on the system requirements. For example, we could define how long an anomalous state should persist before triggering an alarm, or the allowable delay between the twins' behaviors before raising an alert. Additionally, we plan to study the effects of interval period and duration on the alignments and alarm thresholds. Furthermore, we need to analyze the proposal's performance to ensure it is feasible in real-time and verify that the algorithm's computational time supports immediate responses. All these questions need to be investigated and thoroughly discussed with several case studies deployed in different scenarios, which is precisely what we plan to address next. Finally, we are working on a testing framework to evaluate the effectiveness and performance of our proposal in more realistic environments. To this end, we intend to integrate our algorithm into an architecture that facilitates communication with the twins and displays alarms and information via a dashboard. We also plan to compare our approach with similar proposals in the literature, using a set of real-world examples that we are preparing as a benchmark for testing digital twin systems.

Acknowledgments

We want to thank Aitor Arrieta, from Mondragón University, for giving us access to the elevator used in this study and its simulator. This work was partially supported by the Spanish Government (FEDER/Ministerio de Ciencia e Innovación – Agencia Estatal de Investigación) under projects SoCUS [TED2021-130523B-I00] and IPSCA [PID2021-125527NB-I00].

References

[1] Stephen F. Altschul, Warren Gish, Webb Miller, Eugene W. Myers, and David J. Lipman. 1990. Basic local alignment search tool. *Journal of Molecular Biology* 215, 3 (1990), 403–410. https://doi.org/10.1016/S0022-2836(05)80360-2

[2] Manuela Dalibor, Nico Jansen, Bernhard Rumpe, David Schmalzing, Louis Wachtmeister, Manuel Wimmer, and Andreas Wortmann. 2022. A Cross-Domain Systematic Mapping Study on Software Engineering for Digital Twins. *J. Syst. Softw.* 193 (2022), 111361. https://doi.org/10.1016/j.jss.2022.111361

[3] Digital Twin Consortium. 2021. Glossary of Digital Twins. https://www.digitaltwinconsortium.org/glossary/index.htm.

[4] Mohammad Sadegh Es-haghi, Cosmin Anitescu, and Timon Rabczuk. 2024. Methods for enabling real-time analysis in digital twins: A literature review. *Computers & Structures* 297 (2024), 107342.

[5] Opher Etzion and Peter Niblett. 2010. *Event Processing in Action.* Manning Publications.

[6] Cynthia Freeman, Jonathan Merriman, Ian Beaver, and Abdullah Mueen. 2021. Experimental Comparison and Survey of Twelve Time Series Anomaly Detection Algorithms. *J. Artif. Intell. Res.* 72 (2021), 849–899. https://doi.org/10.1613/JAIR.1.12698

[7] Alwyn E Goodloe and Lee Pike. 2010. *Monitoring distributed real-time systems: A survey and future directions.* Technical Report. NASA/CR-2010-216724, Avionics And Aircraft Instrumentation.

[8] Giovanni Lugaresi, Sofia Gangemi, Giulia Gazzoni, and Andrea Matta. 2023. Online validation of digital twins for manufacturing systems. *Comput. Ind.* 150 (2023), 103942. https://doi.org/10.1016/j.compind.2023.103942

[9] Usue Mori, Alexander Mendiburu, and José Antonio Lozano. 2016. Distance Measures for Time Series in R: The TSdist Package. *R Journal* 8, 2 (2016), 451. https://doi.org/10.32614/rj-2016-058

[10] Paula Muñoz, Javier Troya, and Antonio Vallecillo. 2023. A Conceptual Architecture for Building Digital Twins. In *Proc. of MeSS@STAF'23.*

[11] Paula Muñoz, Manuel Wimmer, Javier Troya, and Antonio Vallecillo. 2022. Using trace alignments for measuring the similarity between a physical and its digital twin. In *Proc. of ModDiT@MODELS'22.* ACM, 503–510. https://doi.org/10.1145/3550356.3563135

[12] Saul B. Needleman and Christian D. Wunsch. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Molecular Biology* 48, 3 (1970), 443–453. https://doi.org/10.1016/0022-2836(70)90057-4

[13] Paula Muñoz and Javier Troya and Manuel Wimmer and Antonio Vallecillo. 2024. Measuring the Fidelity of a Physical and a Digital Twin Using Trace Alignments. *Under Review* (2024).

[14] Qinghua Xu, Shaukat Ali, and Tao Yue. 2023. Digital Twin-based Anomaly Detection with Curriculum Learning in Cyber-physical Systems. *ACM Trans. Softw. Eng. Methodol.* 32, 5 (2023), 113:1–113:32. https://doi.org/10.1145/3582571