Journal of Industrial Information Integration 24 (2021) 100242

Contents lists available at ScienceDirect

Journal of Industrial Information Integration

journal homepage: www.sciencedirect.com/journal/journal-of-industrial-information-integration

Developing sensor signal-based digital twins for intelligent machine tools

Angkush Kumar Ghosh [a], AMM Sharif Ullah [b,*], Roberto Teti [c], Akihiko Kubo [b]

[a] Graduate School of Engineering, Kitami Institute of Technology, 165 Koen-cho, Kitami 090-8507, Japan
[b] Division of Mechanical and Electrical Engineering, Kitami Institute of Technology, 165 Koen-cho, Kitami 090-8507, Japan
[c] Department of Chemical, Materials and Industrial, Production Engineering, University of Naples Federico II, Piazzale Tecchio 80, Naples I – 80125, Italy

ARTICLE INFO

Keywords:
Digital twin
Sensor signal
Cyber-physical systems
Machine tool
Monitoring

ABSTRACT

Digital twins can assist machine tools in performing their monitoring and troubleshooting tasks autonomously from the context of smart manufacturing. For this, a special type of twin denoted as sensor signal-based twin must be constructed and adapted into the cyber-physical systems. The twin must (1) machine-learn the required knowledge from the historical sensor signal datasets, (2) seamlessly interact with the real-time sensor signals, (3) handle the semantically annotated datasets stored in clouds, and (4) accommodate the data transmission delay. The development of such twins has not yet been studied in detail. This study fills this gap by addressing sensor signal-based digital twin development for intelligent machine tools. Two computerized systems denoted as Digital Twin Construction System (DTCS) and Digital Twin Adaptation System (DTAS) are proposed to construct and adapt the twin, respectively. The modular architectures of the proposed DTCS and DTAS are presented in detail. The real-time responses and delay-related computational arrangements are also elucidated for both systems. The systems are also developed using a Java™-based platform. Milling torque signals are used as an example to demonstrate the efficacy of DTCS and DTAS. This study thus contributes toward the advancement of intelligent machine tools from the context of smart manufacturing.

1. Introduction

Manufacturing has rapidly been transforming under the umbrella of the fourth industrial revolution, known as Industry 4.0 [1] or smart manufacturing [2]. This transformation diligently utilizes the Information and Communication Technologies (ICT) to bring about automation and autonomy among the manufacturing enablers (e.g., machines, tools, sensors, physical networks among computing devices, actuators, robots, computers, CAD/CAM systems, and alike). As a result, the relevant enablers must perform high-level cognitive tasks, such as monitoring, understanding, predicting, deciding, and adapting in real-time [3,4]. The embedded systems such as Cyber-Physical Systems (CPS) and the Internet of Things (IoT) empower the enablers toward performing the abovementioned high-level tasks. Many authors have embarked on CPS and IoT's implications in Industry 4.0 or smart manufacturing. For the sake of better understanding, some of the recent articles are briefly described below.

Zhou et al. [4] described the conceptual framework of new-generation intelligent manufacturing systems. They proposed a basic mechanism of human-cyber-physical system, consisting of human domain, cyber system, and physical system. The cyber system hosts a self-growing knowledge base that integrates expert knowledge and machine intelligence through new-generation artificial intelligence. Morteza [5] reviewed the extant literature of the Industry 4.0 ecosystem. The entities involved in the ecosystem are simulation and modeling, CPS, semantic technologies, IoT, Internet of Service (IoS), Internet of Data (IoD), cloud computing, big data analytics, blockchain, cybersecurity, augmented reality, automation and industrial robotics, and additive manufacturing. The proximal and distal relationships among these entities achieve personalized product, corporate social responsibilities, smart factory, smart product, interoperability, modularization, decentralization, virtualization, real-time capacity, vertical integration, and horizontal integration. Rossit et al. [6] presented a dynamic scheduling architecture based on data-driven procedures in smart manufacturing environments. Both vertical and horizontal data integration in the CPS to make the dynamic scheduling achievable. The vertical allows establishing direct contact between the physical and decision-making levels and horizontal integrations of CPS-driven business functions traditionally carried out independently. Big data and other relevant technologies become the backbone of dynamic

* Corresponding author.
E-mail address: ullah@mail.kitami-it.ac.jp (A.S. Ullah).

https://doi.org/10.1016/j.jii.2021.100242
Received 12 February 2021; Received in revised form 8 May 2021; Accepted 3 July 2021
Available online 6 July 2021

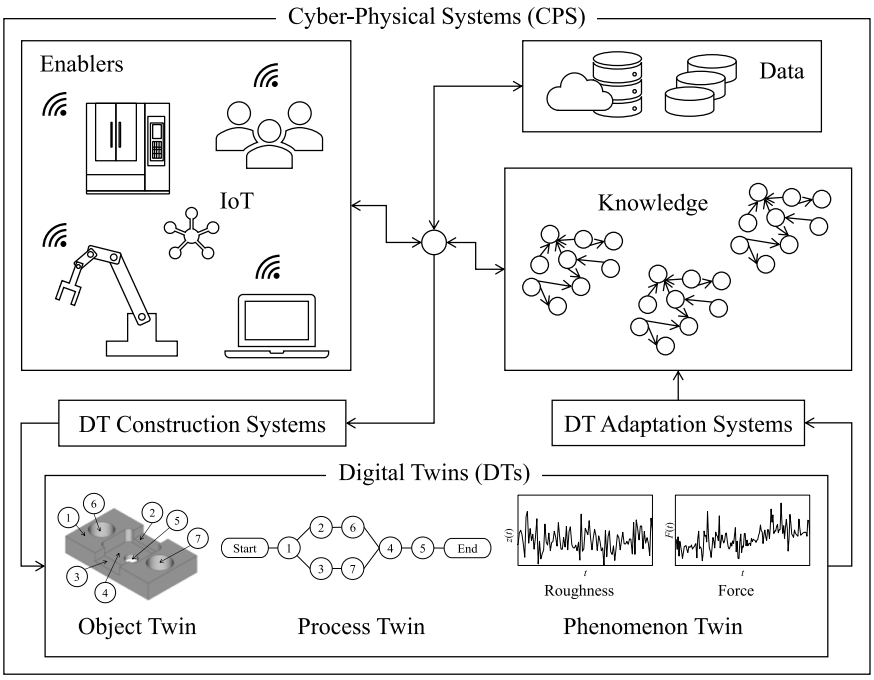


Fig. 1. Role of DTs in CPS [16].

scheduling architecture due to its data dependency. Lu [7] presented a state-of-the-art survey of the ongoing research on Industry 4.0, elucidating its content, scope, and findings. This study emphasizes that the symbiosis of machine-to-machine communication and machine-to-human collaboration will emerge in Industry 4.0. The integration of engineering processes, business processes, and service processes creates a new enterprise information system architecture for Industry 4.0. Oztemel and Gursev [8] provided a literature review on Industry 4.0 and related technologies. They have emphasized the harmonized roles of humans and machines in Industry 4.0—machines perform monotonous and recurrent tasks, opening more opportunities for humans to perform creative tasks. Developing a taxonomy is a pressing need in Industry 4.0. Monostori et al. [9] elucidated the structure, functions, and roles of the CPS in manufacturing science and technology. They emphasized that CPS evolve due to the advent of computer science and information and communication technologies. It will bring all stakeholders and aiding manufacturing systems into a networked type of integration, breaking the hierarchies. Further research, development, and implementation activities are needed to give a concrete shape to manufacturing CPS where the socio-ethical aspects must not be neglected. Yao et al. [10] described the eight-tuple structure of CPS with the characteristics of real-time data access, reconfiguration, interoperation, decentralized decision-making, intelligence, and proactivity to overcome the limitations of existing integrated manufacturing systems. They have also emphasized that the eight-tuple structure must be upgraded to a nine-tuple structure, adding the concept of wisdom manufacturing, incorporating the functionalities of social computing, community, crowdsourcing, customization/personalization, innovation, and sustainability. Lee et al. [11] provided a pragmatic 5C architecture of the CPS in manufacturing consisting of hierarchically organized five layers called connection, conversion, cyber, cognition, and configuration. This architecture helps develop more intelligent and resilient manufacturing equipment and helps achieve better product quality and system reliability. Lu and Cecil [12] outlined the IoT-based collaborative framework, which serves as the foundation for cyber-physical interactions and collaborations in Industry 4.0. They have introduced a language called engineering enterprise modeling language to materialize the exchange of data and information among the IoT embedded devices. Pang et al. [13] developed a data-driven intelligent decision-making method for facilitating quality control in CPS. Different machine learning techniques (e.g., artificial neural network) must be incorporated to perform the quality control tasks in a predictive way. Nakayama et al. [14] described how Industry 4.0 evolves its predecessor (Industry 3.0). They have emphasized that achieving vertical and horizontal integration with the aid of Industrial Intent of Things (IIoT), service systems, semantic web, artificial intelligence, and collaborative networks can transform Industry 3.0 to Industry 4.0. Morgan et al. [15] revisited reconfigurable manufacturing from the context of Industry 4.0. The developed a three-layer modularity framework of cyber-physical and IIoT framework, consisting of physical control, cyber control and artificial intelligence. Physical control (edge) and cyber control (fog) layers are connected by industrial network wherein digital twins play their roles. Cyber control and artificial intelligence (cloud) are connected by enterprise network where services play their roles.

Nevertheless, CPS hosts the constituents of smart manufacturing systems as schematically illustrated in Fig. 1. As seen in Fig. 1, CPS consists of four basic elements, (1) IoT embedded manufacturing enablers, (2) cloud-based data storage and management system, (3) an ever-growing knowledge-base, and (4) arrangements regarding Digital Twins (DTs).

The remarkable thing is that different types DTs [17–19] supply the knowledge for the ever-growing knowledge-base (see Fig. 1). A DT is a computable virtual abstraction of a real-world entity, which must respond in real-time [17]. There are three types of DT: (1) object twin, (2) process twin, and (3) phenomenon twin [16]. Here, an object twin means the DT of a real-world object (e.g., workpiece, cutting tool, machine tool, sensor, and alike) used in a given manufacturing environment. A process twin means the DT of a process sequence in a given manufacturing environment. A phenomenon twin means the DT of a machining phenomenon (e.g., surface roughness, cutting force, tool wear, cutting torque, and alike) in a given manufacturing environment. These twins collectively recreate the real-world manufacturing environment in the cyber space, supply the relevant pieces of knowledge, and drive the abovementioned high-level cognitive tasks (monitoring, understanding, predicting, deciding the right course of actions, and

adapting). Now, DTs are not readily available. They must be constructed. As such, two systems denoted as DT Construction System (DTCS) and DT Adaptation System (DTAS) must exist—DTCS constructs the relevant DTs and DTAS uses constructed DTs for functionalizing the cognitive tasks.

Many authors have been embarked on DTs' conceptualization, systematization, construction, and adaptation from the context of Industry 4.0 and beyond. DT can be understood with respect to digital model and shadow, as described by Aheleroff et al. [17]. The digital model has no real-time connectivity with its physical counterpart, but the digital shadow has limited or one-directional connectivity with its physical counterpart. On the other hand, DT has bi-directional real-time connectivity with its physical counterpart. It has been stressed that DT needs semantically annotated content for fulfilling its bidirectional real-time connectivity. There is another relevant concept known as a digital thread. This concept is synonymous with the digital model or digital shadow where the model is product-life-cycle-aware, that is, the model is semantically annotated in terms of its role in the product-life-cycle. However, many authors have studied DTs. For example, Fuller et al. [20] reviewed the recent developments regarding DTs from different viewpoints (manufacturing, healthcare, and smart cities). They described that a DT incorporates data analytics, artificial intelligence (AI)-based modeling, simulation, and visualization for decision-making. They suggested that the DT might include validation measures to ensure the trustworthiness of the AI-generated models before putting them into practice. Kaur et al. [21] mentioned that a DT machine learns from real-time sensory data for forecasting the future of the corresponding physical counterparts. Jiang et al. [22] described that the DTs drive value-added services (intelligent operation and maintenance, monitoring, and optimization of assets) in IoT embedded industrial environment. For this, they proposed a data-model-service (DMS)-based framework for developing the DTs. The Industrial Internet Consortium (IIC) [23] articulated that the DTs can be designed discretely (relevant to a single entity in a real-life manufacturing environment) for monitoring purposes. The discrete DTs can also be composed to create a composite DT for realizing the overall manufacturing environment. Nevertheless, the DTs acquire data (physical objects' data, time-series data, historical data, and alike) for modeling physical entities. The models represent the behavior of those entities by incorporating machine learning and simulation techniques. The IIC highlighted that the DTs might contain a set of human-comprehensive interfaces for putting them (DTs) into practice in industrial applications and making them available to other stakeholders whenever required. Tao et al. [24] have presented a DT-driven approach for product development. The approach incorporates information related to developing a product (design, design requirements, process, historical data in the form of big data, and other associated factors such as environmental factors, market review, and customer feedback). It improves the product design and optimizes relevant production and process plans. Ruppert et al. [25] described that the real-time sensory data-driven DTs can be integrated with the CPS for monitoring the status of the enablers and relevant decision-making. They also introduced a DT framework called real-time locating systems (RTLS)-based DT for production scheduling. Aheleroff et al. [26] stressed the role of DT for Mass Personalization Manufacturing (MPM). They described that DT promises a high degree of personalization with the aid of IoT, artificial intelligence (AI), additive manufacturing (AM), and cloud-based technologies in the smart manufacturing paradigm. They also proposed a three-dimensional DT reference model consisting of DT architecture, agile product development, and DT integration hierarchy.

However, DTs can make a machine tool intelligent [27–29]. As a result, DT-embedded machine tools monitor and troubleshoot their activities autonomously. For this purpose, sensor signal-based DTs (phenomenon twins) must be constructed and adapted into the CPS apart from other types of DTs. Sensor signal-based DTs can be built in many ways. One of the pragmatic ways is to consider the issue of data transmission delay [30,31]. Sensor signals are subjected to delay while being transmitted among different systems in the CPS. When this delay is considered, the signal analyses (e.g., time and frequency domains-based analyses) may not adequately capture the dynamics of the underlying phenomenon. Instead, a domain called the delay domain [32,33] can be considered for capturing the dynamics of the underlying phenomenon. This is explained in detail in the next section, where an arbitrary signal is analyzed in time, frequency, and delay domains, showing that the delay domain-based analysis is perhaps the most pragmatic approach for capturing the dynamics of the underlying phenomenon. Therefore, sensor signal-based DTs built based on delay domain are important for developing intelligent machine tools. There have been fewer studies in this direction. This study fills this gap. Based on the above consideration, this study proposes a DTCS for constructing delay-dependent sensor signal-based DTs and a DTAS for adapting the constructed DTs in a real-life manufacturing environment. The proposed DTCS and DTAS are developed using a Java™-based platform, and their efficacy is also investigated. Thus, the rest of this article is organized as follows. Section 2 presents the analyses of an arbitrary signal in time, frequency, and delay domains and demonstrates the importance of delay domain from the context of sensor signal-based DTs. Section 3 presents the architecture of the DTCS and DTAS from the viewpoint of intelligent machine tools. Section 4 presents the developed DTCS and DTAS, and demonstrates their efficacy. Section 5 provides the concluding remarks of this study.

2. Delay and its implication

As mentioned in the previous section, sensor signals are subjected to delay while being transmitted among different systems in the CPS. When this delay is considered, the signal analyses (e.g., time and frequency domains-based analyses [34]) may not adequately capture the dynamics of the underlying phenomenon. Instead, a domain called the delay domain [32] can be considered for capturing the dynamics of the underlying phenomenon. Based on this consideration, the first half of this section describes some of the selected studies on delay-based signal processing. The last half of this section presents the implication of delay using an arbitrary signal, proving some guidelines showing how to accommodate the delay in constructing sensor signal-based DT.

2.1. Delay-related studies

Many researchers active in communication and control engineering have been working on delay from the context of Industry 4.0-relevant communication networks, focusing on its impacts and compensation mechanisms. Some of the recent articles are briefly described below.

Raptis et al. [35] reviewed the data delivery delay and its compensation for the sake of data management (coordination and computation of data) in Industry 4.0. Zoppi et al. [36] developed a Quality of Service (QoS) framework for mitigating end-to-end delay in industrial wired/wireless sensor networks (WSN). Mo et al. [37] focused on the performance of the networked control systems (NCS) due to delay. They showed how the delay impedes the reliability of NCS. Zhang et al. [38] proposed a delay compensation algorithm for the NCS using a method called Controller Area Network (CAN)-buses. Guck et al. [39] introduced a Software-defined Networking (SDN)-based function split framework for reducing delay. It (SDN-based framework) meets the end-to-end delay requirements, guaranteeing the real-time QoS of data exchange. Yagi and Sawada [40] articulated that delay underlying a communication network is random. It causes instability in the feedback systems. They also designed a Kalman filter to reduce the effect of the random delay. Fan et al. [41] articulated that a random delay underlying the NCS degrades the control performance, making the system unstable. In this regard, they presented a control scheme called Networked Predictive Control (NPC). It mitigates the effects of transmission delay and achieves desired control performance using two components: Network

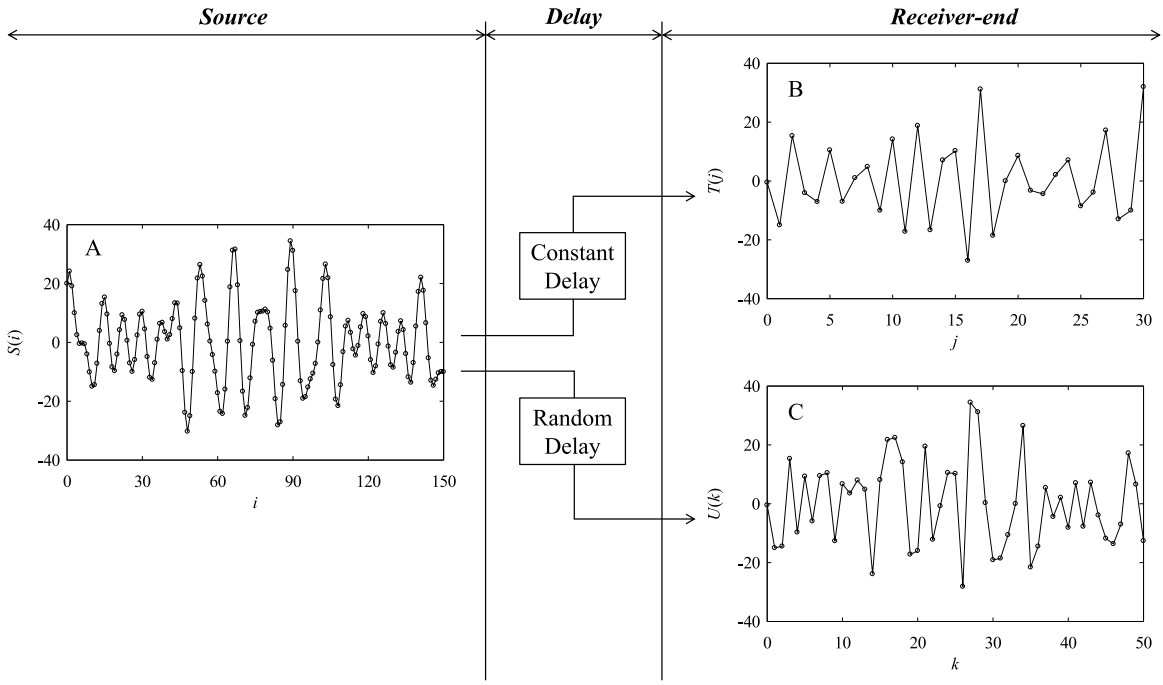


Fig. 2. Implication of delay in $S(i)$.

Delay Compensator (NDC) and Control Prediction Generator (CPG), respectively. Wu et al. [42] described sensor-to-actuator and controller-to-actuator delays using Markov chains. They also proposed a control scheme that compensates these delays and makes the NCS stable. Sun and Huo [43] developed a switching control approach called Markovian Jump Linear System (MJLS). It models the random delays as a Markov process and makes the NCS stable. Guo and Gu [44] described that random time delay is the cause of instability and poor performance in the NCS. They also suggested a model to improve stability. Baillieul and Antsaklis [45] referred to delay as unavoidable and one of the challenges of modern networked control systems composed of heterogeneous systems and applications. They also mentioned that the sensors might fail to transmit data immediately, and data loss may occur due to communication delays. They considered that distributed control systems perform better in overcoming delay-related issues compared to the centralized ones. Bijami and Farsangi [46] described that communication delay—underlying distributed networked systems composed of heterogeneous sub-systems—causes random data loss and makes the system unstable. They also developed a control framework for stabilizing the networked systems. Zunino et al. [47] mentioned that delay compensation is needed to transmit data at a higher update rate for materializing the real-time behavior of Industry 4.0. They also emphasized that systems and applications in Industry 4.0 need to be delay-aware for addressing the time-related performance and reliability requirements. Ferrari [31] identified different types of delays in an IoT-based environment, such as end-to-end delay, round-trip-time, and jitter. These delays cause low data delivery rate, data loss, and failure of the control systems. The author provided a hardware-independent methodology for measuring delays in an industrial IoT-based environment. Jhaveri et al. [30] presented a delay-aware framework to measure the end-to-end delay in real-time SDN-based networks. Kontogiannis and Kokkonis [48] proposed a protocol called Fuzzy Real-Time haPticS protocol (FRTPS) for alleviating the impacts of delay in real-time applications. It (FRTPS) alleviates data loss and low data delivery for achieving better synchronization and reliability of the systems. Xia et al. [49] proposed an algorithm called Mixed-Criticality Relative-execution Deadline (MCRD) for managing delay, scheduling of data transmission, and reducing data loss in the industrial internet of things (IIoT)-based environments. Basir et al. [50] mentioned that fog computing performs better than cloud computing for compensating delay-related issues. It also achieves low latency data delivery and reliable real-time communication among heterogeneous systems in the IIoT-based environments. They also mentioned that the systems and applications in the IIoT need to be delay-aware to limit different types of delays (e.g., processing, propagation, transmission, and computation delay) while achieving real-time communication. Wang et al. [51] emphasized that communication delay needs to be considered while designing controllers to monitor and control in the NCS.

2.2. Implication of delay in signal processing

As described above, delay causes data loss and system instability in a given communication and data transmission network. This is true for the networks in CPS. Accordingly, this section presents the implication of delay using an arbitrary signal, proving some guidelines showing how to accommodate the delay in constructing sensor signal-based DT.

Consider an arbitrary signal denoted as $S(i) \mid i = 0,1,\ldots$ that follows Eq. (1). In Eq. (1), $a_1,\ldots,a_4$ and $f_1,\ldots,f_4$ are the amplitude and frequency components, respectively, where $a_1 = 15$, $a_2 = 10$, $a_3 = 5$, $a_4 = 10$, $f_1 = 0.08$, $f_2 = 0.03$, $f_3 = 0.06$, and $f_4 = 0.045$. As seen in Fig. 2, $S(i)$ (denoted as A) is the source signal. When $S(i)$ is transmitted from the source and received at a receiver-end, how delay impacts it ($S(i)$) is the research question here. Say, a constant delay is occurred in this process, denoted as c, where $c = 5$. As a result, the receiver-end receives a constant delay-driven signal denoted as $T(j)$, where $T(j) = S(i + c) \mid j = 0,1,\ldots$, as shown in Fig. 2 (denoted as B). On the other hand, say, a random delay is occurred, denoted as r_l, where $r_{l=1,\ldots,5} \in \{1,2,3,4,5\}$. As a result, the receiver-end receives a random delay-driven signal denoted as $U(k)$, where $U(k) = S(i+r_l) \mid k = 0,1,\ldots$, as shown in Fig. 2 (denoted as C).

$$S(i) = a_1\sin(2\pi i f_1) + a_2\cos(3 \times 2\pi i f_2) + a_3\sin(2\pi i f_3) + a_4\cos(3 \times 2\pi i f_4) \quad (1)$$

As seen in Fig. 2, when a delay occurs whether it is constant or random, the characteristic of the source signal (here $S(i)$) gets affected. However, from visual inspection, the time series of the signals ($S(i)$, $T(j)$ and $U(k)$) are not so insightful for understanding the underlying

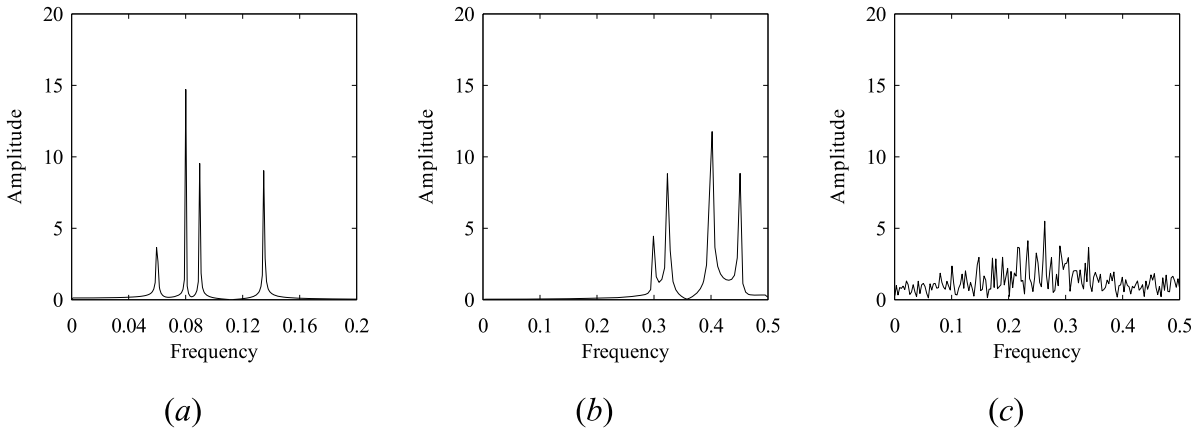


Fig. 3. Frequency domain representations (FFT) of: (*a*) *S*(*i*), (*b*) *T*(*j*), and (*c*) *U*(*k*).

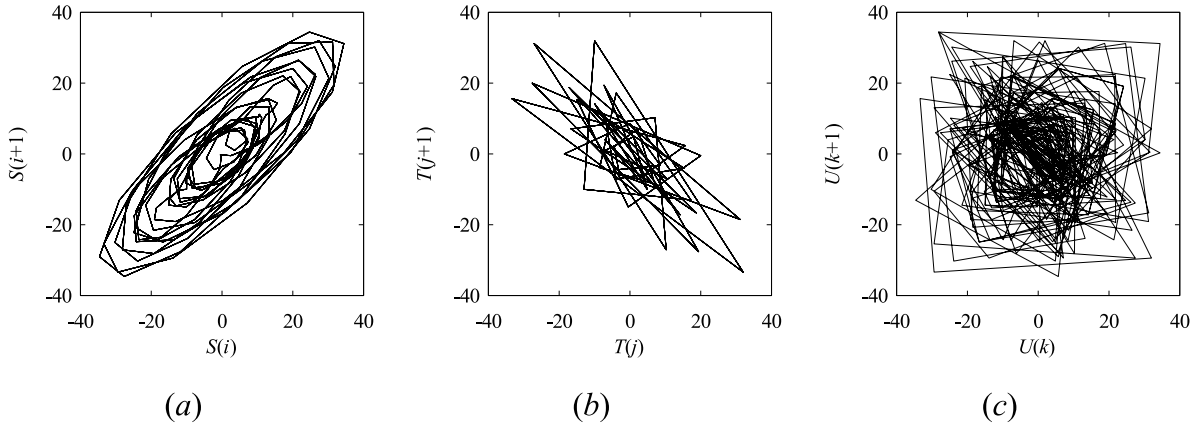


Fig. 4. Delay maps of: (*a*) *S*(*i*), (*b*) *T*(*j*), and (*c*) *U*(*k*).

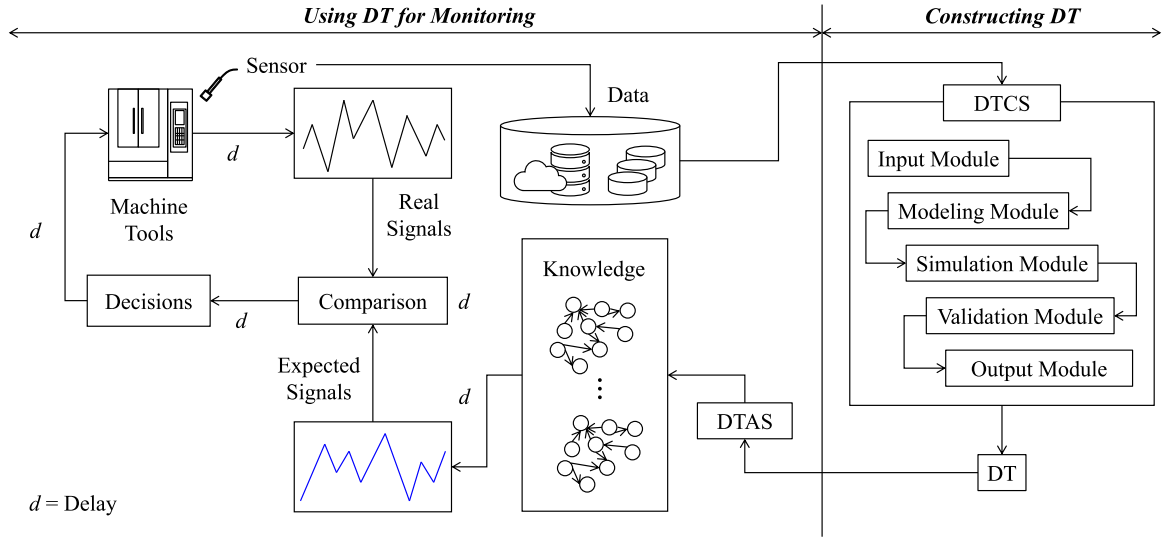


Fig. 5. Construction and use of DT in intelligent machine tools.

dynamics. For this reason, the abovementioned signals (*S*(*i*), *T*(*j*), and *U*(*k*)) are processed in the frequency domain (using Fast Fourier Transformation (FFT)). The corresponding results are shown in Fig. 3(*a*)–(*c*), respectively. As seen in Fig. 3(*a*), the frequency domain represents the frequencies underlying *S*(*i*) as expected. As seen in Fig. 3(*b*), the frequencies underlying *T*(*j*) are not the same compared to that of *S*(*i*) (see Fig. 3(*a*)). This means that the constant delay can change the frequency information associated with the source signal, *S*(*i*). As seen in Fig. 3(*c*), the frequencies underlying *U*(*k*) are jumbled. This means that the random delay can change the frequency information associated with the source signal, *S*(*i*) (see Fig. 3(*a*)), significantly.

In addition, the abovementioned signals (*S*(*i*), *T*(*j*), and *U*(*k*)) are

Table 1
Functional requirements of the modules underlying the DTCS.

Modules	Fundamental Purposes	Functional Requirements							
		Ontology	Semantic Web Compatibility	Real-time Response	Data Management	Modeling	Machine Learning	Simulation	Validation
Input	Signal Acquisition	○	○	○	○				
Modeling	Knowledge Extraction			○		○	○		
Simulation	Signal Simulation			○				○	
Validation	Characteristic Comparison			○					○
Output	Transferring DT to DTAS	○	○	○	○				

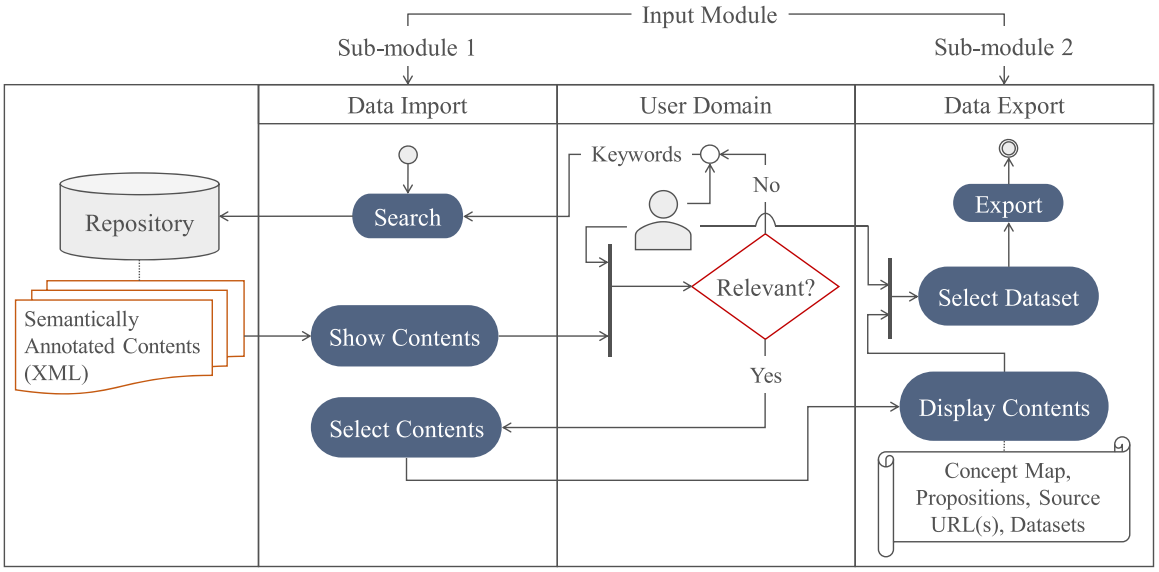


Fig. 6. Modular architecture of the Input module.

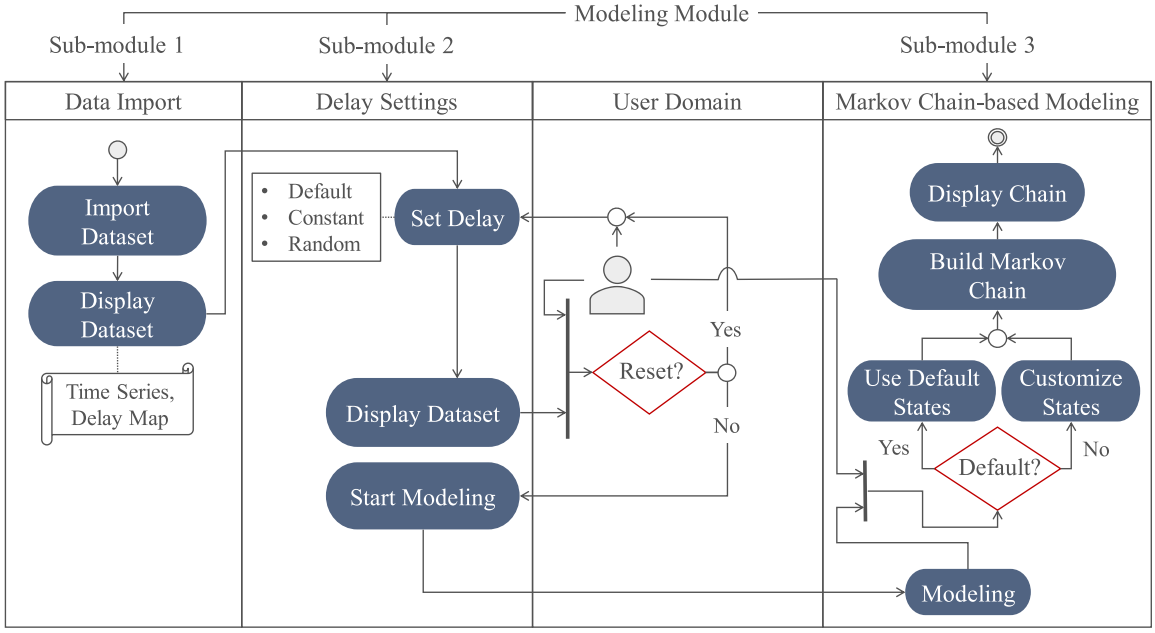


Fig. 7. Modular architecture of the Modeling module.

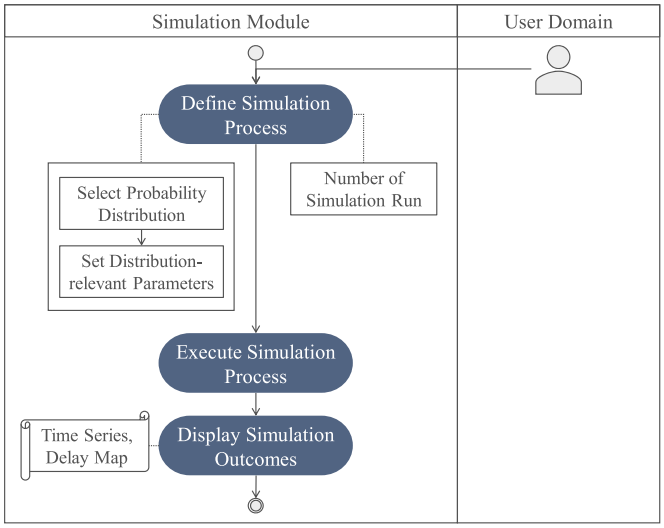


Fig. 8. Modular architecture of the Simulation module.

transferred to the delay domain, which is often recommended for chaotic signals. The corresponding results are shown in Fig. 4(*a*)−4(*c*), respectively. Fig. 4(*a*) shows the delay map of $S(i)$ consisting of points $(S(i),S(i+1))$. It is seen that the returns from one point to another follow a very systematic pattern. This means $S(i)$ is not chaotic. As seen in Fig. 4(*b*), the delay map of $T(j)$ consisting of points $(T(j),T(j+1))$, is somewhat distorted compared to that of $S(i)$ (see Fig. 4(*a*)). This means that the constant delay has affected the systematic behavior of the $S(i)$. As seen in Fig. 4(*c*), the delay map of $U(k)$ consisting of points $(U(k),U(k+1))$, is distorted compared to that of $S(i)$ (see Fig. 4(*a*)) and $T(j)$ (see Fig. 4(*b*)). The returns from one point to another does not follow a systematic pattern anymore. This means that the random delay makes the signal chaotic.

The abovementioned example shows that the dynamics underlying a signal becomes complex due to delay, particularly due to random delay. In such cases, the delay domain is more informative than the frequency domain. However, in CPS, a random delay is most likely to be associated with the sensor signals. As a result, while constructing and adapting the sensor signal-based DTs, the relevant systems (DTCS and DTAS) must accommodate delay domain-based signal processing technique. Based on this consideration, the following section presents an architecture of DT for intelligent machine tools, where the sensor signal is processed in the delay domain subjected to a random delay.

3. DT-driven intelligent machine tools

This section presents an architecture of sensor signal-based DT for intelligent machine tools. The proposed architecture is schematically illustrated in Fig. 5. As seen in Fig. 5, the interplay of a DT and a machine tool undergoes two phases. In the first phase, the DT is constructed using the DTCS. In the other phase, the DTAS injects the constructed DT into the knowledge base. As seen in Fig. 5, the injected DT produces outcomes (a simulated sensor signal) whenever necessary. On the other hand, the machine tool produces a sensor signal while operating. These two signals (simulated and real) are compared to decide the course of action while performing the intelligent machine tool's monitoring and troubleshooting activities. As described before, when the DT is in the use phase (in this case monitoring phase, as shown on the left-hand side in Fig. 5), it is subjected to delay as denoted by "d" in Fig. 5. Therefore, the DT must be constructed based on a delay-embedded signal processing method. Otherwise, the comparison does not make any sense. Based on this contemplation, this study proposes a DTCS and DTAS. For better understanding, this section first presents the proposed DTCS and its functional requirements. Afterward, the modular architectures of the DTCS and DTAS are presented in detail.

3.1. DTCS and its functional requirements

As seen on the right-hand side in Fig. 5, the DTCS consists of five

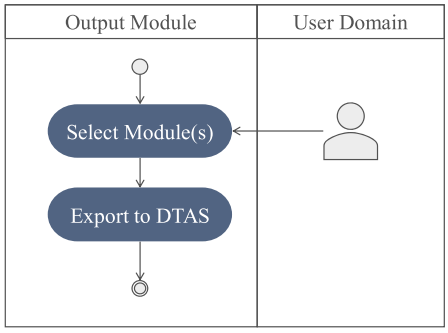


Fig. 10. Modular architecture of the Output module.

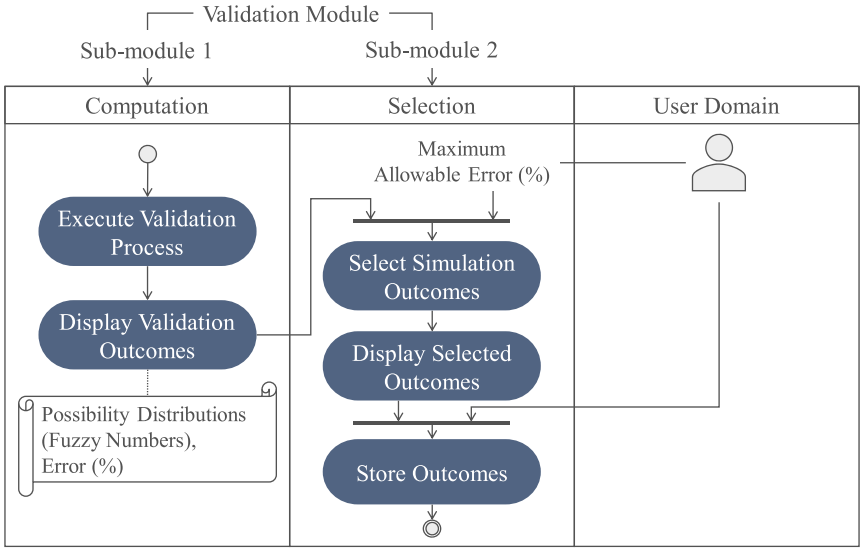


Fig. 9. Modular architecture of the Validation module.

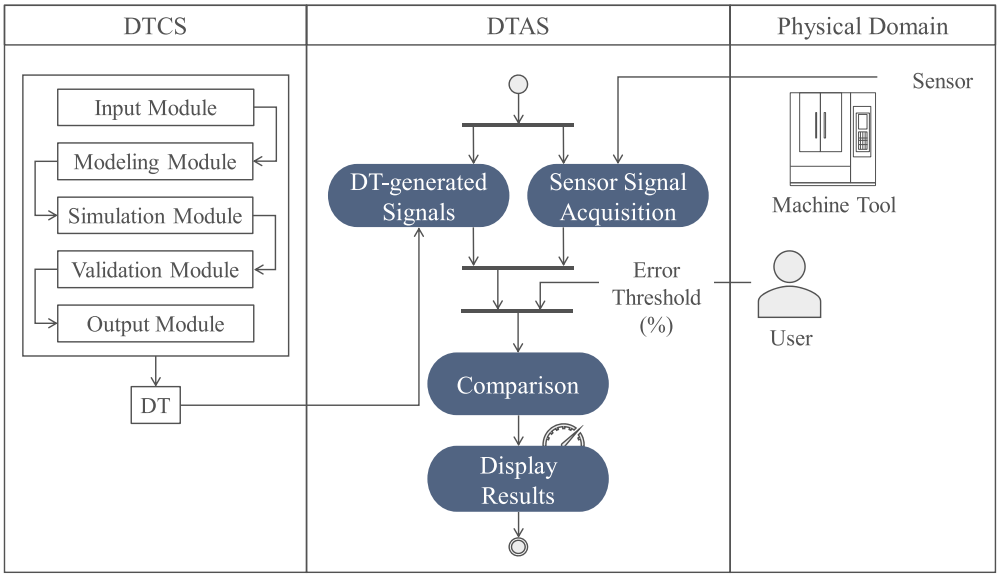


Fig. 11. Modular architecture of the DTAS.

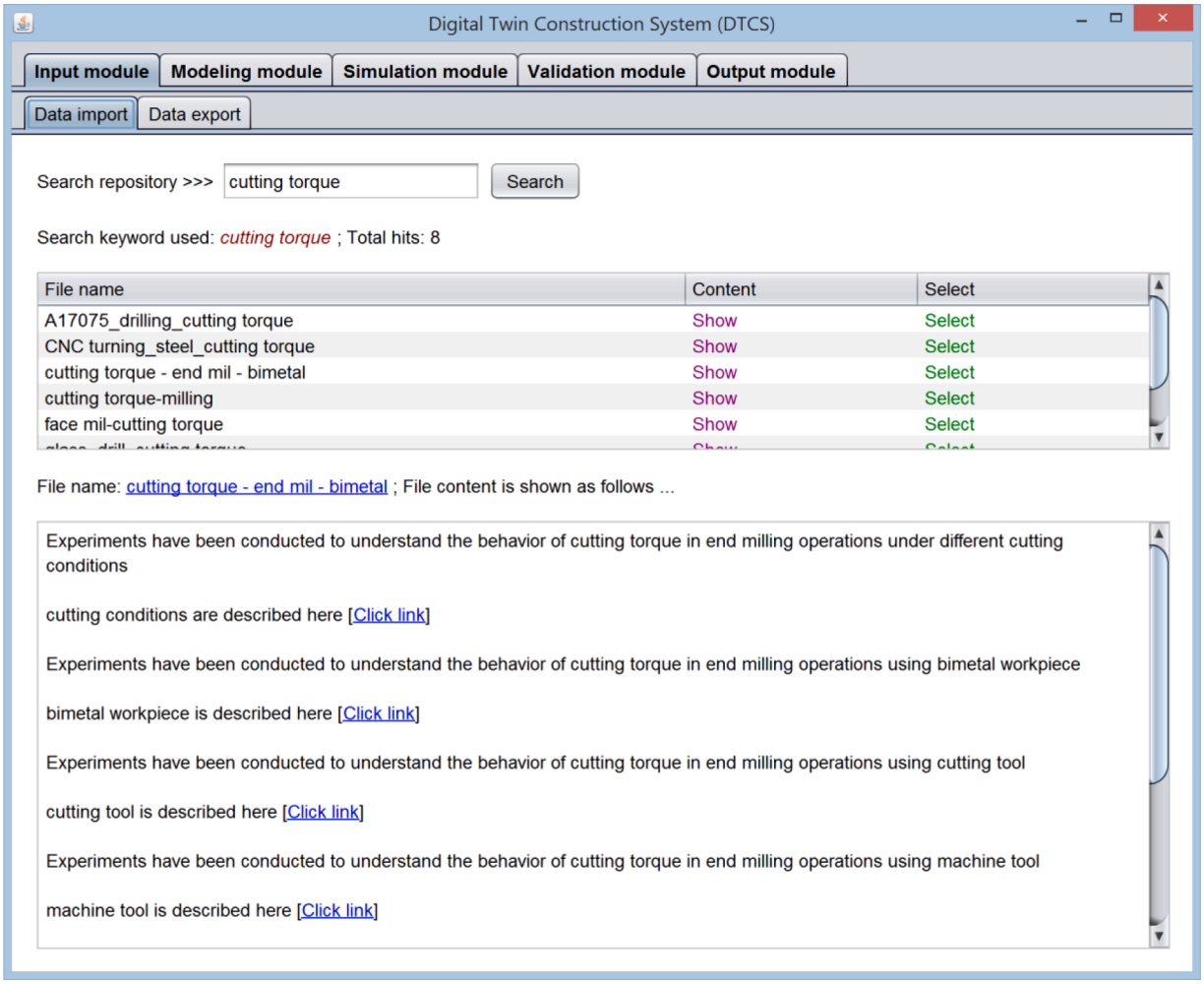


Fig. 12. Screen-print of Data import submodule.

modules: (1) Input, (2) Modeling, (3) Simulation, (4) Validation, and (5) Output modules [16]. These modules collectively constitute the DT. The fundamental purpose of these modules are: (1) Acquiring the right piece of sensor signal data from a given data repository, (2) Extracting knowledge underlying the acquired sensor signal, (3) Simulating sensor signal(s) relevant to the acquired one based on the extracted knowledge,

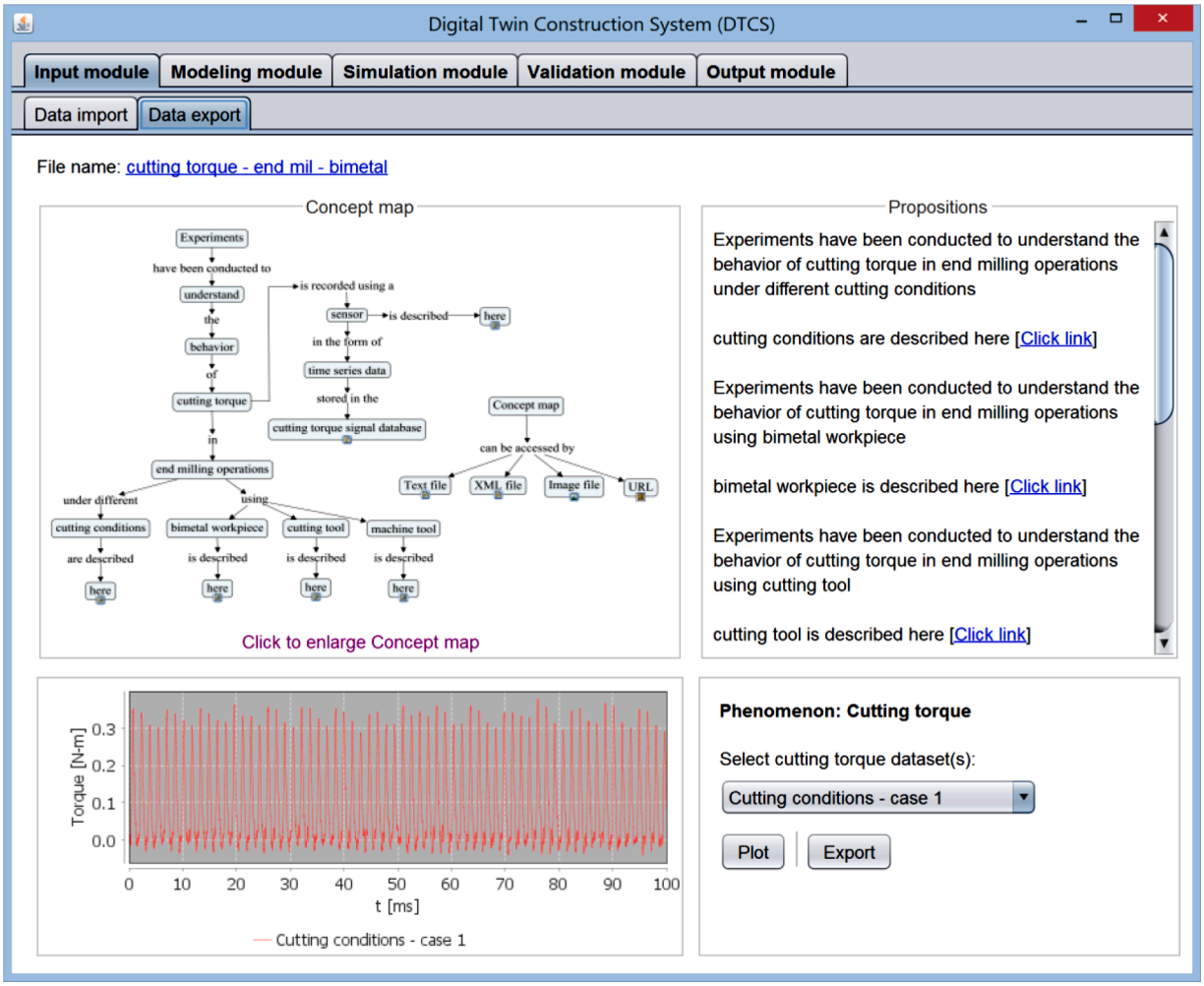


Fig. 13. Screen-print of Data export submodule.

(4) Comparing the characteristics between the real (acquired) and simulated signals, and (5) Transferring the constructed DT to the DTAS, respectively. For this, these modules collectively carry out the functional requirements called Data Management, Ontology, Machine Learning, Modeling, Simulation, Validation, Real-time Response, and Semantic Web Compatibility. Note that the function called Data Management means to collect and manage data from a data storage system (e.g., cloud-based data storage system, see Figs. 1 and 5). Ontology means the high-level descriptions of the datasets and associated entities (e.g., machining process, machining conditions, sensor, sensor signal, workpiece, cutting tool, and machines) in the form of user-defined semantic annotations, making the contents compatible with semantic web and concept maps [52,53]. Modeling means to model the relevant phenomenon, making the machine learning from sensor signal possible. Machine Learning means to learn rules for simulation using a suitable machine learning approach (e.g., neural networks, deep learning, DNA-based computing, and Markov chain). Simulation means to simulate a signal on demand using a suitable approach (e.g., discrete event-based Monte Carlo simulation and deterministic simulation). Validation means to validate the simulation outcomes using a suitable approach (e.g., possibility distribution [54], entropy [55], DNA-based computing [56], Decisional DNA [57], and phenomenon-dependent parameters [16]), ensuring the trustworthiness of the simulation process. Real-time Response means reconfiguring the relevant modules responding to real-time signal data updates [17,27]. Semantic Web Compatibility means to make the whole process of digital twining compatible with the semantic web [58], which will dominate Industry 4.0 and beyond. The relationship among the abovementioned functional requirements and the modules of the DTCS are shown in Table 1.

As seen in Table 1, Real-time Response is associated with all the modules. Apart from it (Real-time Response), the Input module is associated with the Ontology, Semantic Web Compatibility, and Data Management functional requirements. Similarly, the Modeling module is associated with the Modeling and Machine Learning functional requirements. The Simulation module is associated with the Simulation functional requirement. The Validation module is associated with the Validation functional requirement. The Output module is associated with the Ontology, Semantic Web Compatibility, and Data management functional requirements. If the modules can execute the abovementioned functional requirements, then the DT is said to be constructed properly. This means that relationships shown in Table 1 are the design guidelines for constructing a DT.

Recall the functional requirement called Real-time Response, which is involved with all the modules. In particular, this functional requirement is subjected to delay. Therefore, it (delay) needs careful consideration. Otherwise, when the DT is in use (see the left-hand side scenario in Fig. 5), it may not produce the desired outcome. Nevertheless, the modular architectures of the abovementioned DTCS and DTAS are presented, as follows.

3.2. Modular architecture of DTCS

As mentioned in the previous section, the proposed DTCS consists of five modules: Input, Modeling, Simulation, Validation, and Output

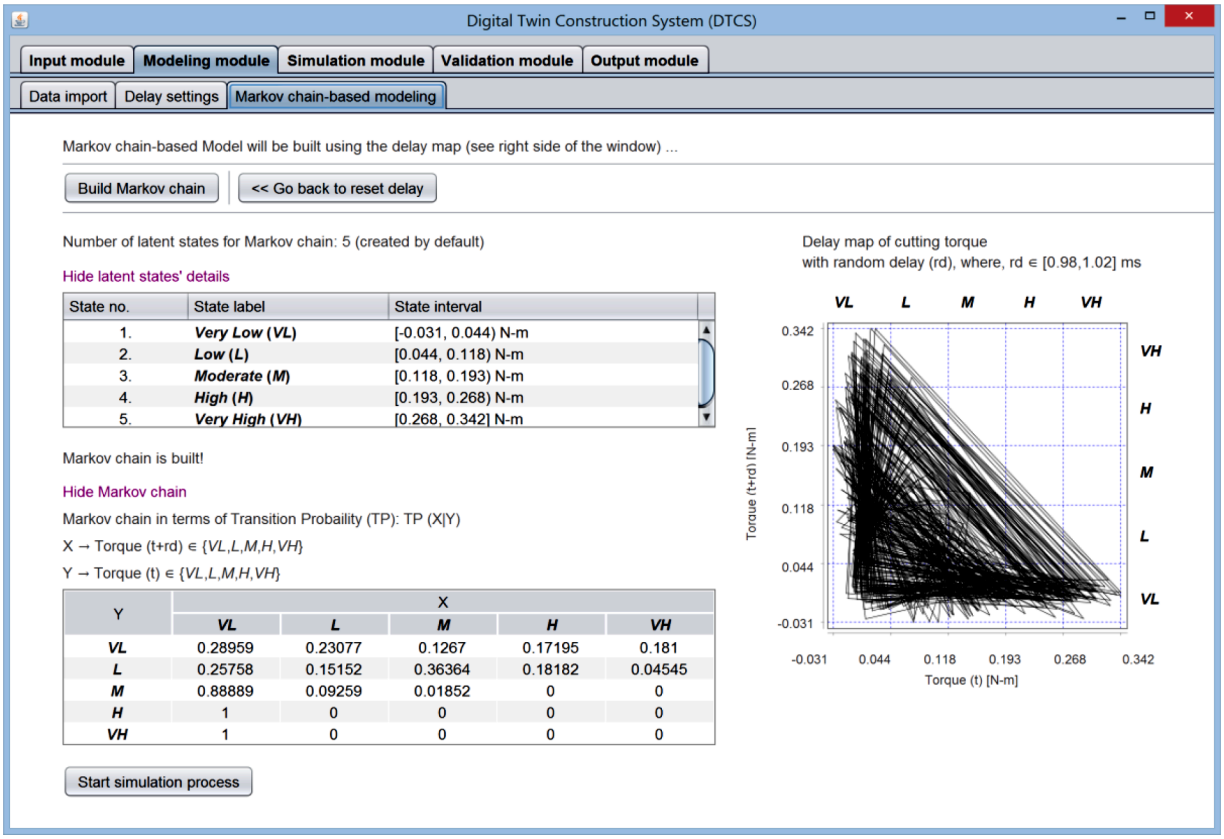


Fig. 14. Screen-print of Modeling module.

modules. These modules collectively construct the DT. The modular architecture of these modules is presented as follows.

3.2.1. *Input module*

The Input module is the first module of the proposed DT. The associated functional requirements are Ontology, Semantic Web Compatibility, Data management, and Real-time Response. Therefore, the Input module must deal with semantically annotated contents. The ideal case would be to link it to a semantic web-embedded data repository. In reality, this may not be the case. As such, the Input module must be linked to an ontology-embedded repository. Besides, it must respond to any content update in real-time. Since the goal here is to create a DT based on sensor signal data, the Input module must provide a facility to search and select the relevant sensor signal data.

Based on the above consideration, a repository is created where signal data of different machining phenomena (e.g., cutting force, cutting torque, surface roughness, and alike) are stored using Extensible Markup Language (XML) file format. The contents are semantically annotated where the relevant concepts (e.g., machining process, cutting conditions, sensor, sensor signal datasets, cutting tool, and alike) and their relationships are defined. Thus, the content takes the form of concept maps. The relevant node of each concept map contains the numerical datasets of a phenomenon. The Input module interacts with the repository using two submodules, denoted as Data import and Data export submodules.

Fig. 6 schematically illustrates the modular architecture of this module. As seen in Fig. 6, the Data import submodule imports semantically annotated contents from the repository based on a user-defined keyword and delivers the contents to the Data export submodule. For this, it operates based on search-show-select functions. The Data export submodule exports the imported contents to the desired location (e.g., Modeling module). For this, it operates based on display-select-export functions.

3.2.2. *Modeling module*

The Modeling module is the second module of the proposed DT. The associated functional requirements are Modeling, Machine Learning, and Real-time Response. It (Modeling module) first acknowledges the signal data (exported by the Input module) and then models the underlying phenomenon using a predefined machine learning approach. One of the approaches (e.g., neural networks, deep learning, semantic modeling, DNA-based computing, or Markov chain) can be used to create a model. Besides, the module must rebuild the model responding to any update in the exported signal in real-time.

Since the delay is an inherent characteristic of the systems where the DT is supposed to work (see Fig. 5), a model created from a given delay map of a signal is perhaps the best option. As such, the Modeling module uses Markov chain-based modeling approach since it highly correlates with delay maps [59,60].

Fig. 7 schematically illustrates the modular architecture of this module. As seen in Fig. 7, the Modeling module consists of three submodules, denoted as Data import, Delay Settings, and Markov chain-based modeling submodules. The Data import submodule imports the dataset exported by the Input module, and displays its (dataset) time series and delay map. The Delay settings submodule allows the user to set a delay out of three choices (default, constant, random), and thereby, constructs delay-driven dataset. It also displays the delay-driven dataset (time series and delay map) for the sake of user comprehensibility. As such, the user may reset the delay if required. The Markov chain-based modeling submodule extracts knowledge underlying the delay map (obtained from the Delay settings submodule) in the form of a Markov chain. For this, the user may use a set of default or customized latent states. Note that the relevant mathematical formulations for Markov chain-based modeling are beyond the scope of this study. Refer to Ref. [61] for the details.

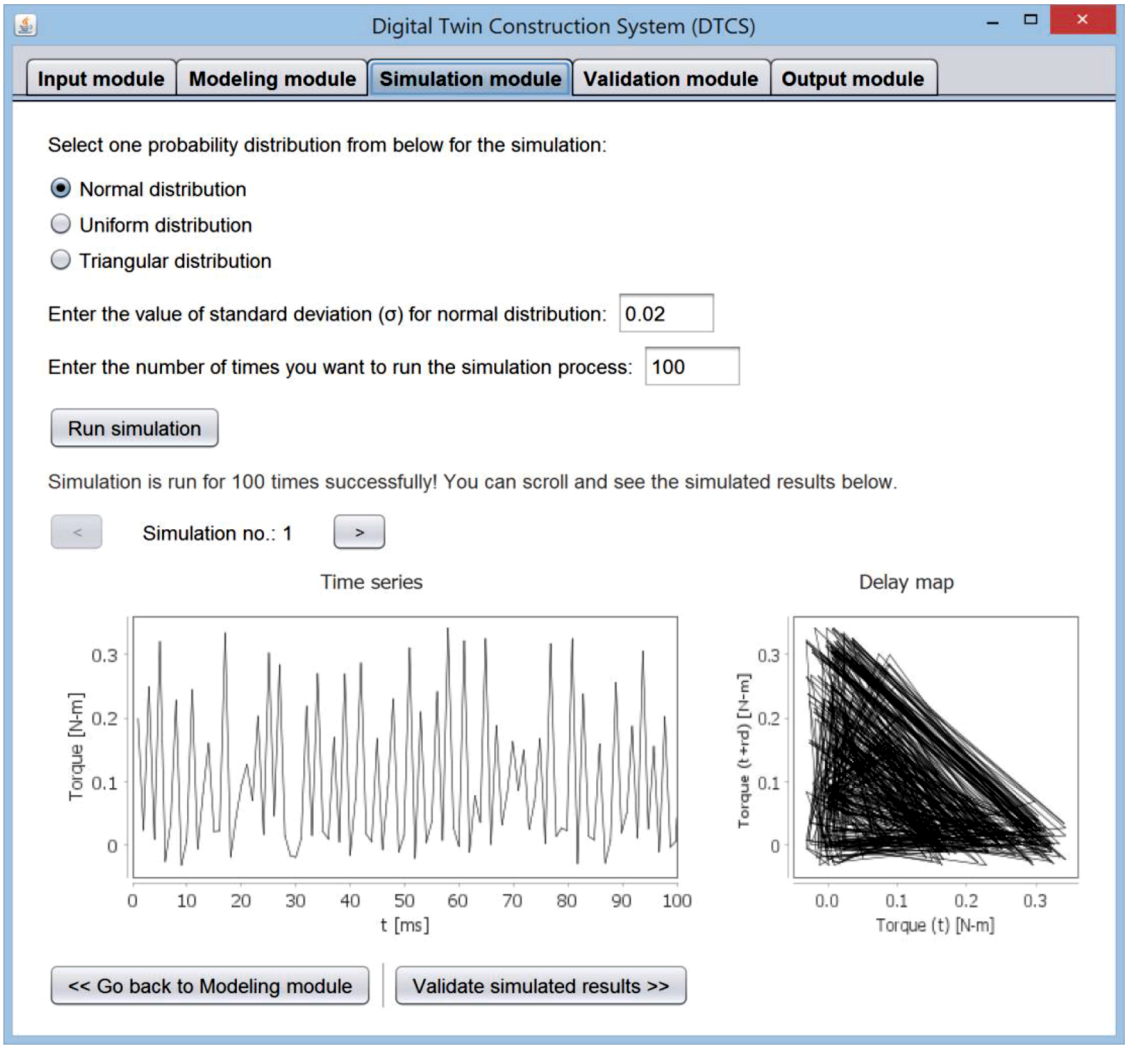


Fig. 15. Screen-print of Simulation module.

3.2.3. *Simulation module*

The Simulation module is the third module of the proposed DT. The associated functional requirements are Simulation and Real-time Response. It (Simulation module) first acknowledges the model created by the Modeling module and then simulates the phenomenon using a predefined simulation approach (discrete event-based Monte Carlo simulation or deterministic simulation). Besides, the module must respond to the model update on a real-time basis. Since the Modeling module uses the Markov chain, the Simulation module uses a discrete event-based Monte Carlo simulation approach. The module first simulates the latent states (states defined in the Modeling module) and translates the states into their numerical counterpart based on a predefined probability distribution. Note that the relevant mathematical formulations and simulation algorithms are beyond the scope of this study. One may refer to the work described in [61] for the details.

Fig. 8 schematically illustrates the modular architecture of this module. As seen in Fig. 8, the module operates based on define-execute-display functions. The user defines the Monte Carlo simulation approach by selecting a probability distribution and setting the distribution-relevant parameters. The user also decides how many times the same simulation process should run. Based on these, the module executes the simulation process and displays the simulation outcomes in time series and delay maps.

3.2.4. *Validation module*

The Validation module is the fourth module of the proposed DT. The associated functional requirements are Validation and Real-time Response. It (Validation module) verifies the simulation outcomes' appropriateness, using one or more pre-defined validation approaches (e.g., possibility distribution, entropy, DNA-based computing, Decisional DNA, and phenomenon-dependent parameters). It then selects and stores some of the validated simulation outcomes for re-use. Besides, the module must respond to any update in the simulation outcomes on a real-time basis.

Fig. 9 schematically illustrates the modular architecture of this module. As seen in Fig. 9, the module consists of two submodules, denoted as Computation and Selection submodules. The Computation submodule is responsible for executing the validation process. In this study, it (Computation submodule) uses possibility distribution (fuzzy numbers) [54] for the sake of validation. Note that other parameters can be integrated if required. On the other hand, the Selection submodule is responsible for selecting and storing some of the validated outcomes based on some user inputs.

3.2.5. *Output module*

The Output module is the last one. The associated functional requirements are Ontology, Semantic Web Compatibility, Data management, and Real-time Response. It (Output module) acts as the connecting point between the DTCS and DTAS. As such, it must interact with the

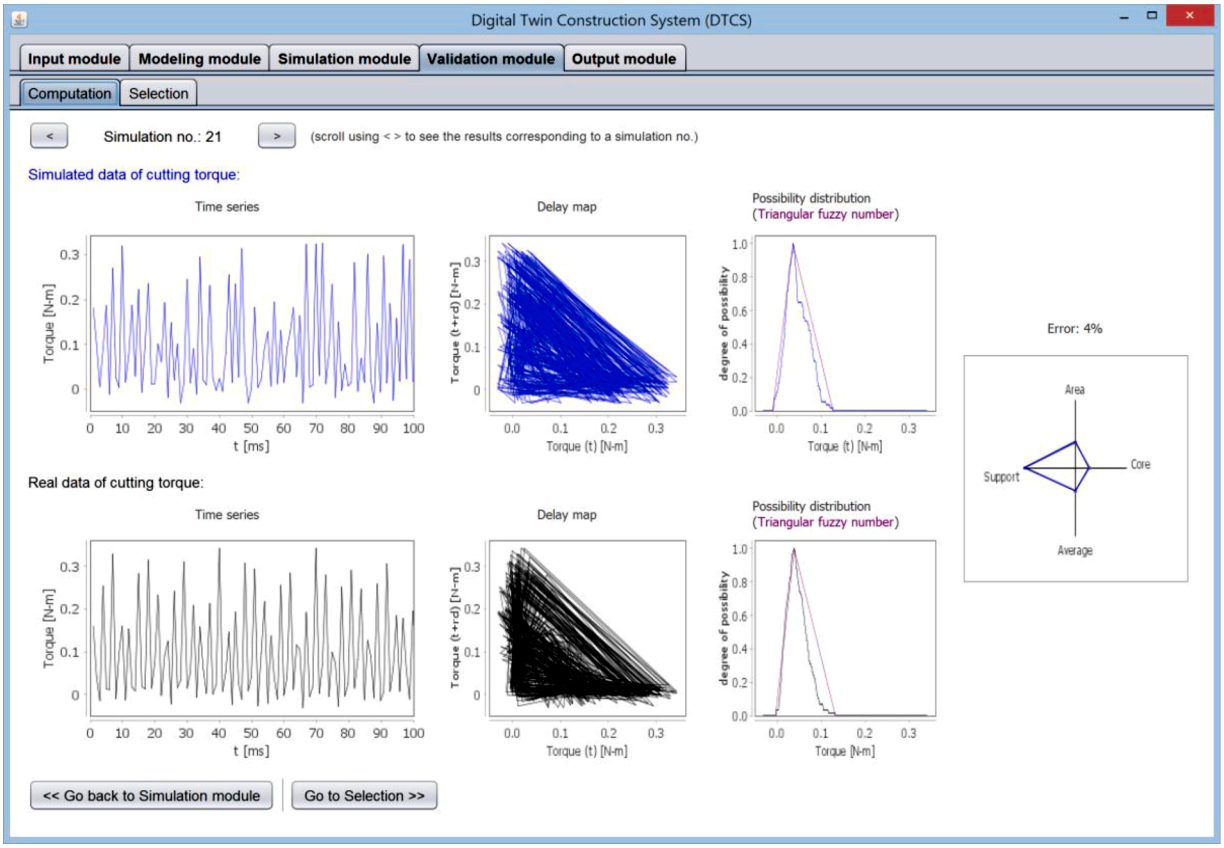


Fig. 16. Screen-print of the Computation submodule.

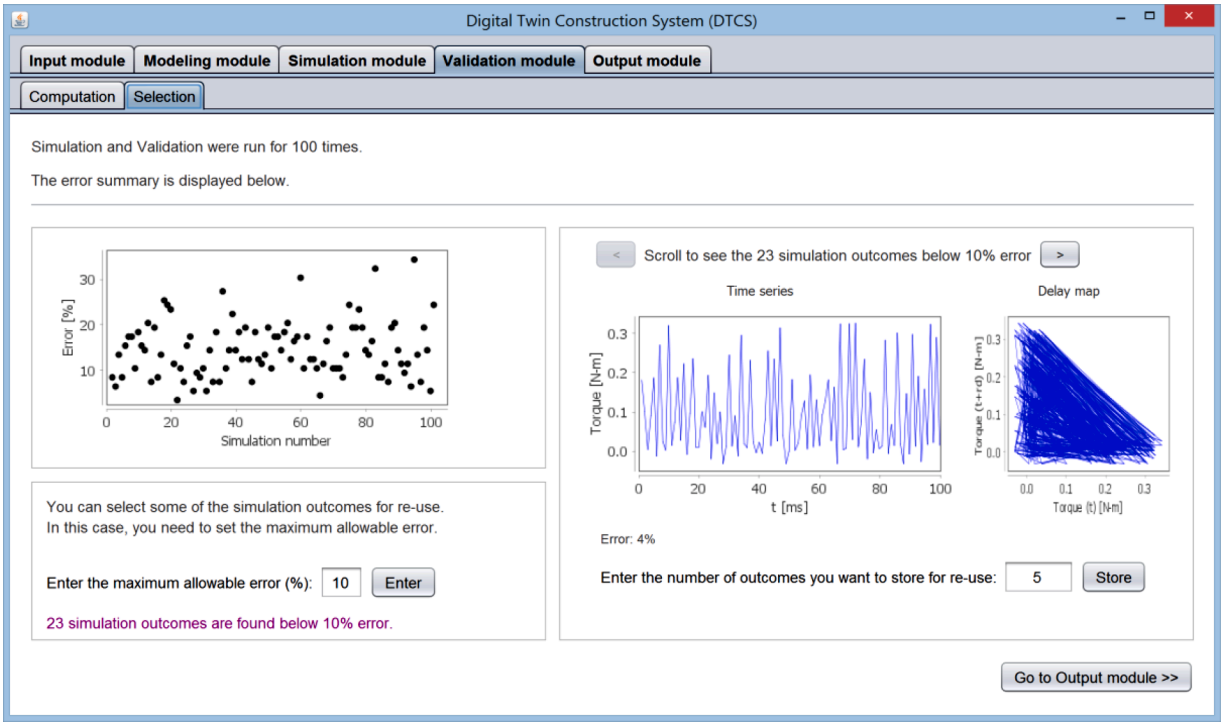


Fig. 17. Screen-print of the Selection submodule in the Validation module.

other modules (Input, Modeling, Simulation, and Validation modules), acknowledge their contents, and export them to the DTAS whenever required. Besides, it must respond to any update in any of the modules mentioned above in real-time.

Fig. 10 schematically illustrates the modular architecture of this module. As seen in Fig. 10, the module operates based on select-export

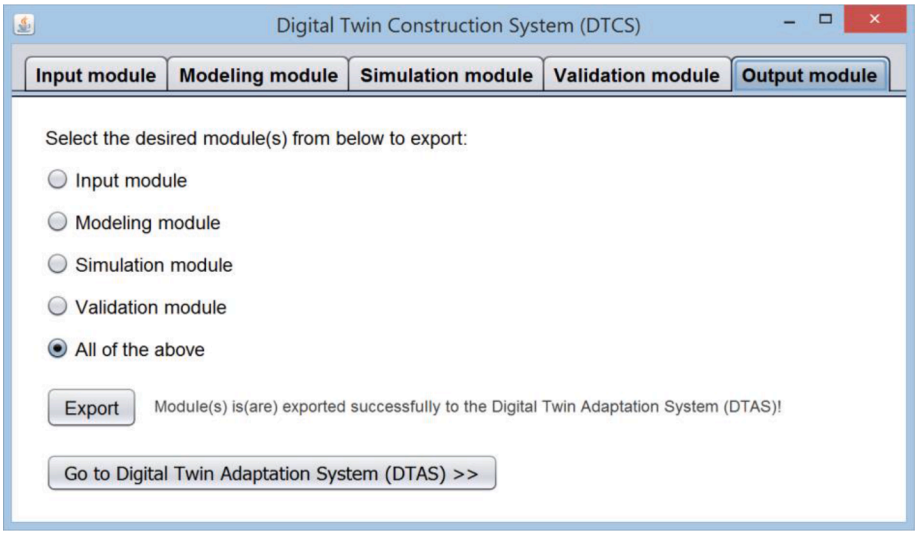


Fig. 18. Screen-print of Output module.

functions. As such, the module transfers the constructed DT to the DTAS whenever required.

3.3. Modular architecture of DTAS

The DTAS uses the constructed DT for monitoring a real-life machining process. Fig. 11 schematically illustrates the modular architecture of the DTAS. As seen in Fig. 11, it acknowledges the DT-generated signals and uses the signals to monitor a real-life machining process in real-time. For this, it uses a user-defined error threshold (%). It displays the monitoring results so that the user can decide the right course of action.

Based on the abovementioned architectures, the DTCS and DTAS are developed using a Java™-based platform, as presented in the following section.

4. Constructing DT

This section presents a DT that is constructed using a Java™-based platform. The constructed DT consists of two integrated systems (DTCS and DTAS) presented in the following two sections, respectively. The scenario here is to monitoring a machining process. In addition, the efficacy of the constructed systems is also described.

4.1. DTCS

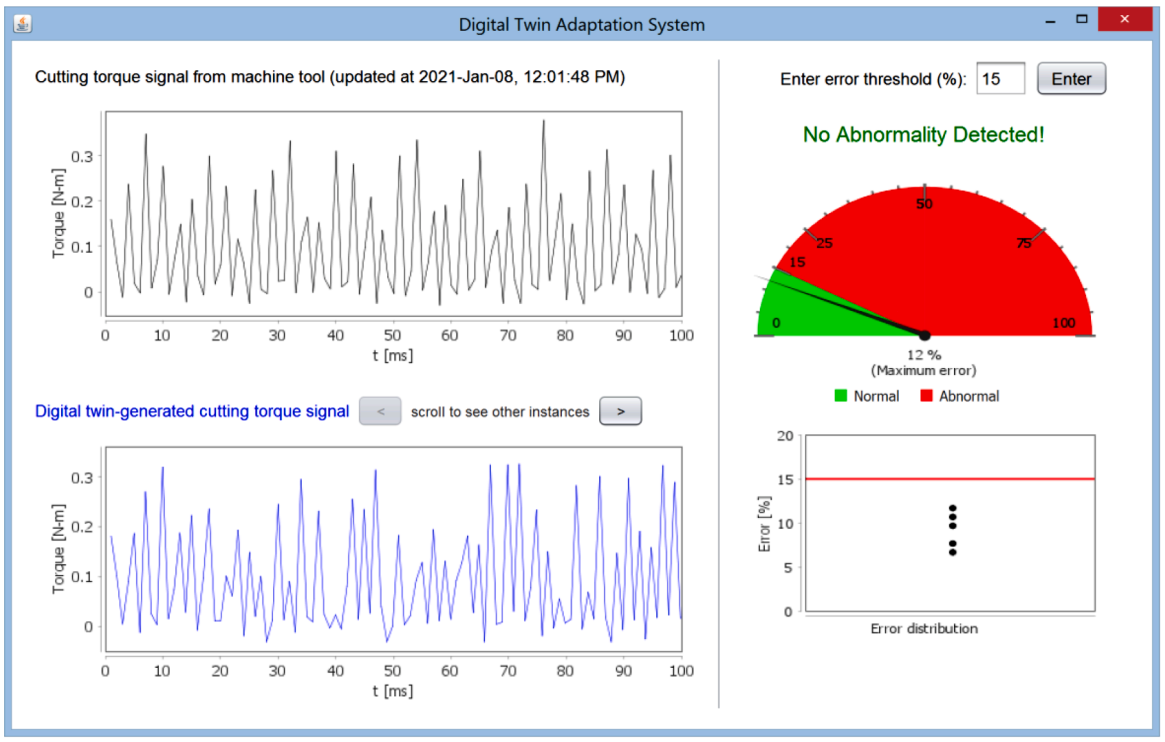
Consider a machining phenomenon called cutting torque created due to a machining process (e.g., milling). In this case, the user can use the Data import submodule of the Input module (Fig. 12) to search semantically annotated contents based on a keyword (here, cutting torque). The submodule provides a facility ('Show' button in Fig. 12) to visualize the fetched content. It also provides a facility ('Select' button in Fig. 12) to import the desired content and deliver it to the Data export submodule of the Input module.

Fig. 13 shows the user interface of the Data export submodule. It first displays the imported content in the form of a concept map, a list of propositions, and graphical plots of numerical datasets. The submodule then provides a facility ('Plot' button in Fig. 13) to visualize and select the datasets. It also provides a facility ('Export' button in Fig. 13) to export the selected dataset to the Modeling module or any other data storage systems. One of the remarkable aspects of this Input module is its scalability with user-defined ontology. For example, the contents shown in Fig. 13 (concept map and underlying propositions) can be represented in various ways in the repository. The module can respond to such variability. This means that the module does not depend on strict ontological formalism. Rather, it provides a flexible way of annotating machine and human-readable content.

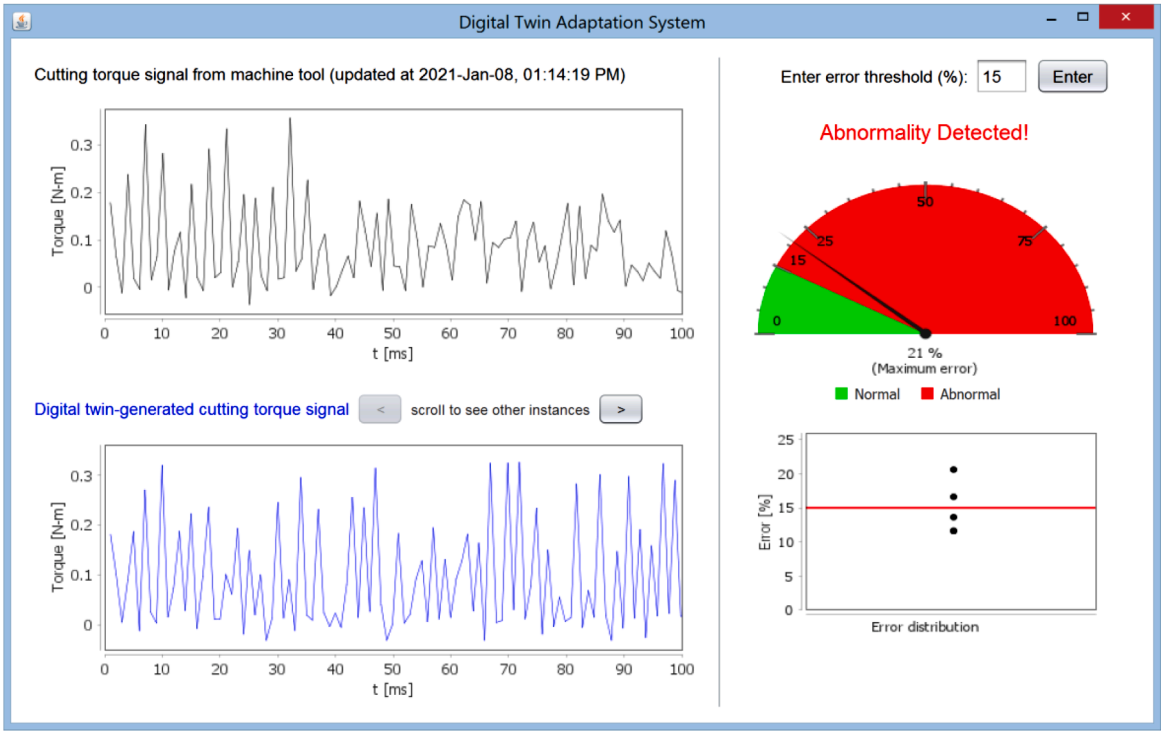
Fig. 14 shows the user interface of the Modeling module when a Markov chain is created from a delay map of a cutting torque signal (imported from the Input module, see Fig. 13). The map corresponds to a random delay, that is, 1 ± 0.02 ms. Five latent states, namely *Very Low* (*VL*), *Low* (*L*), *Moderate* (*M*), *High* (*H*), and *Very High* (*VH*) are used. The transition probabilities of these states are calculated and displayed. As such, the transition matrix serves as the model (Markov chain) of the given torque signal. If a user prefers, the other labels and the number of latent states can be used for modeling. The preferences can be set using the user interface (not visible in Fig. 14).

Fig. 15 shows the user interface of the Simulation module. Here the user sets a probability distribution to simulate the numerical counterparts of latent states. The user can choose a probability distribution out of normal, uniform, triangular distributions and set the corresponding parameters. In addition, the user can set the number of simulation instances. In the case shown in Fig. 15, the user has chosen the normal distribution and set the standard deviation equal to 0.02. Other values can be used if preferred. In addition, the user has set the number of simulation instances equal to 100. As such, the system runs the same simulation process 100 times and creates 100 sets of torque signals. The simulation outcomes (time series and delay map) are shown in the user interface. The user can scroll the time series and delay maps to see the consistency of the simulation process.

Fig. 16 shows the user interface of the Computation submodule of the Validation module. It displays the time series and delay map used to construct the model in the Modeling module (black-colored plots). It also displays the simulated time series and delay map created in the Simulation module (blue-colored plots). (In Fig. 16, the simulated time series and delay map correspond to simulation number 21.) Whether or not the simulated outcomes correspond to the real one, the Computation submodule induces possibility distributions (fuzzy numbers) [54], quantifying the uncertainty in the real and simulated outcomes. The submodule displays the corresponding possibility distributions for the visual inspection. The user can scroll the number of simulations and see the corresponding results. In the graphs of possibility distributions, the induced triangular fuzzy numbers are also displayed (purple-colored curves). Apart from visual inspection, four parameters, denoted as Area, Average, Core, and Support, are used to quantify the similarity between the possibility distributions, as shown on the right-hand side in Fig. 16. This results in a single error measure, defined as Error (see Appendix A). The instance shown in Fig. 16 corresponds to the Error of 4%.



(*a*)



(*b*)

Fig. 19. Use of constructed DT in monitoring: (*a*) no abnormality detected, (*b*) an abnormality detected.

Fig. 17 shows the user interface of the Selection submodule of the Validation module. It first displays the error summary in the form of a scatter plot between the Error and the number of simulations (shown on the left-hand side in Fig. 17). It then provides a facility to select some of the simulation outcomes based on a user-defined maximum allowable Error. The case shown in Fig. 17 corresponds to the maximum allowable Error of 10%. As such, the submodule retrieves 23 simulation outcomes and displays corresponding results in the form of time series and delay maps (shown on the right-hand side in Fig. 17). The user can scroll the retrieved outcomes and see the corresponding results. The user can store

all or some of the retrieved outcomes in a repository for reuse. In the case shown in Fig. 17, five (5) of the retrieved simulation outcomes are stored. As such, these five simulation outcomes can be exported to other stakeholders (e.g., DTAS) on-demand via the Output module.

Fig. 18 shows the user interface of the Output module. The module exports the other module(s) to the DTAS based on user selection. In the case shown in Fig. 18, all the modules are exported. This means that all the five torque simulation results (available in the Validation module) and the relevant contents (available in other modules) are exported to the DTAS. Therefore, the DTAS utilizes those simulation results for monitoring a real-life machining process, as presented in the next section.

4.2. DTAS

Fig. 19 shows the user interface of the DTAS. The DTAS acknowledges all the five DT-generated cutting torque signals (visible one signal at a time) and uses them in monitoring. As seen in Fig. 19(*a*), the interface displays the time series of one of the DT-generated torque signals (blue-colored plot) and the torque signal from a machine tool (black-colored plot). The user can scroll to see the other DT-generated torque signals. The system measures the errors between the signal from the machine tool and the DT-generated signals to detect an abnormality. For this, the DTAS considers a user-defined error threshold. In the case shown in Fig. 19(*a*), an error threshold of 15% is used. As seen on the top right-hand side in Fig. 19(*a*), the maximum error (12%) lies under the threshold (green-colored region). This means that the torque signal from the machine tool shows no abnormality. Apart from the maximum error, the error distribution related to all the DT-generated signals can also be seen in the form of a scatter plot, as shown on the bottom right-hand side in Fig. 19(*a*). On the other hand, the system detects abnormality when any of the error values exceed the threshold. This scenario is shown in Fig. 19(*b*). Note that the error computation follows the mathematical formulations described in Appendix A.

5. Concluding remarks

Futuristic machine tools need the assistance of the DTs to perform their monitoring and troubleshooting tasks. These twins are not readily available. Systems are needed to construct and adapt these DTs. Two computerized systems are developed on the Java™-based platform, denoted as DTCS and DTAS. The first system constructs a DT, and the other adapts the constructed DT.

The DTCS consists of five modules: Input, Modeling, Simulation, Validation, and Output modules. These modules collectively construct the DT. These modules' functional requirements are Data Management, Ontology, Machine Learning, Modeling, Simulation, Validation, Real-time Response, and Semantic Web Compatibility.

The Input module can search semantically annotated datasets stored in a remote data storage facility and select the appropriate signal datasets. This module handles both human-comprehensible contents (concept maps) and machine-readable contents (XML format). This arrangement ensures effective data mining because signal datasets accompany other relevant information (e.g., machining process, machine tools, machining conditions, sensors, and alike), making it meaningful to different stakeholders.

The Modeling module machine learns the dynamics underlying the Input module-supplied sensor datasets. In this case, numerous machine learning methods can be used. In this article, a Markov chain-based machine learning method is used. This method works well when the signal datasets are sampled from a delay map, acknowledging the underlying delay (random or deterministic). Delay is unavoidable, as described in Section 2. The Modeling module supplies the machine-learned model of the signal datasets to the Simulation module. This module simulates the signal using the modeling information received. In this case, a discrete event stochastic simulation process is recommended, as shown in Section 4. Whether or not the simulation module has simulated the signal datasets faithfully can be tested in the Validation module. Thus, the Valuation module is equipped with some quantitative measures. In this article, a possibility distribution-based approach is used for validation. This method works well, as shown in Section 4. The Output module transfers the contents generated in other modules for reuse. The user can select the contents of the Input, Modeling, Simulation, and Validation modules for transferring them to a data storage facility in the XML format. One of the default locations is DTAS. Thus, the Output module is the connecting point between the DTCS and DTAS. Though the DTAS can adapt all the contents generated in DTCS, it uses only the simulated signal datasets tested positive by the Validation module while monitoring a process. It receives real-time signals from a machine tool for monitoring purposes. Therefore, it is equipped with a user interface showing the monitoring results, as shown in Section 4. Any update in the DTCS will change the contents. Therefore, DTAS also updates itself, acknowledging the changes made in DTCS in real-time. These real-time updating capabilities make these two systems highly coupled.

In this article, milling torque signals are used as an example. Other signals also show promising results. Thus, the DTCS and DTAS can be used to ensure machine tools' ability to perform their monitoring and troubleshooting tasks autonomously. This way, the finding of this study contributes to the advancement of Industry 4.0 or smart manufacturing.

Nevertheless, a DT can improve the cyber-physical integration of industrial IoT. In most cases, the twin must machine-learn the required knowledge from the historical sensor signal datasets and seamlessly interact with the real-time sensor signals. While doing so, the DTs must handle the semantically annotated datasets stored in clouds and accommodate the data transmission delay. The presented DTCS and DTAS fulfill these requirements of DTs. Therefore, these can be used as general tools for information integration in smart manufacturing. Further research can be conducted in this direction.

CRediT authorship contribution statement

Angkush Kumar Ghosh: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **AMM Sharif Ullah:** Conceptualization, Data curation, Funding acquisition, Investigation, Methodology, Project administration, Resources, Software, Supervision, Validation, Writing – original draft, Writing – review & editing. **Roberto Teti:** Conceptualization, Formal analysis, Writing – original draft, Writing – review & editing. **Akihiko Kubo:** Conceptualization, Data curation, Formal analysis, Validation, Writing – original draft, Writing – review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Parameters to quantify a possibility distribution (fuzzy number)

Let $\{x(t) \in \Re \mid t = 0, \Delta t, 2\Delta t, \ldots, n \times \Delta t\}$ be a real sensor signal collected from a manufacturing process. Here, Δt is the delay. As such, the delay map of the real signal consists of the ordered pairs $\{(x(t),x(t+\Delta t)) \mid t = 0, \Delta t, 2\Delta t, \ldots, n\text{-}1 \times \Delta t)\}$. Let $\{s(t) \in \Re \mid t = 0, \Delta t, 2\Delta t, \ldots, n \times \Delta t\}$ be a DT generated signal (i.e., simulated signal). As such, the delay map of the simulated signal consists of the ordered pairs $\{(s(t),s(t+\Delta t)) \mid t = 0, \Delta t, 2\Delta t, \ldots, n\text{-}1 \times \Delta t)\}$. From a given return map, a possibility distribution (fuzzy number) can be induced following the mathematical formulations described in [54]. Thus, the point cloud of the real signal induces a possibility distribution denoted as $\text{Poss}(x) \in [0,1]$ and a triangular fuzzy number (TFN), as schematically illustrated in Fig. A1(a). The core of the TFN is one of the cores of $\text{Poss}(x)$. The support of the TFN is the support of $\text{Poss}(x)$. Similarly, the point cloud of the simulated signal induces a possibility distribution denoted as $\text{Poss}(s) \in [0,1]$ and a triangular fuzzy number (TFN), as schematically illustrated in Fig. A1(b).

Now, a set of four parameters denoted as P_i, $i = 1,\ldots,4$, can be considered to quantify a possibility distribution and a TFN. Here, P_1 is the area under the possibility distribution, and P_2 is the average possibility, as schematically illustrated in Fig. A2. On the other hand, P_3 and P_4 are the core and the

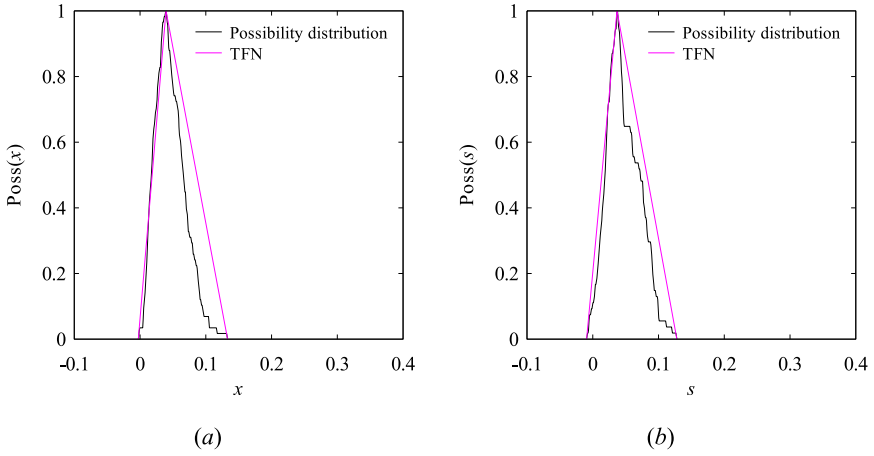


Fig. A1. Possibility distributions and TFNs of: (a) real signal and (b) simulated signal.

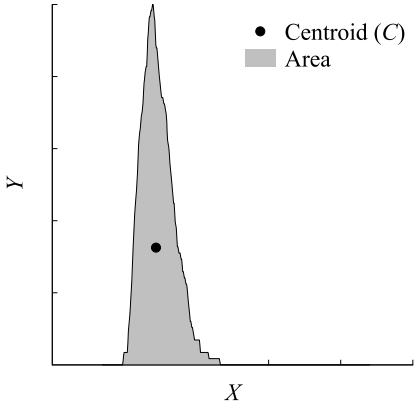


Fig. A2. P_1 and P_2 from a given possibility distribution.

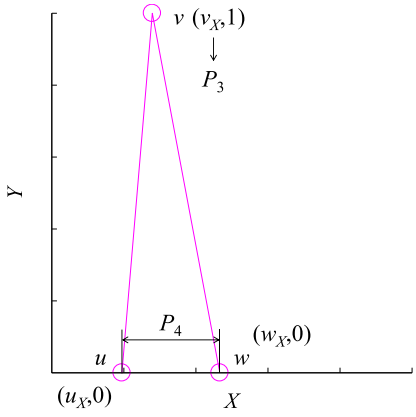


Fig. A3. P_3 and P_4 from a given TFN.

range calculated from the support of TFN, respectively, as schematically illustrated in Fig. A3. In Figs. A2, A3, $\exists X \in \{x, s\}$ and $\exists Y \in \{\text{Poss}(x), \text{Poss}(s)\}$.

Let $\{(X_j, Y_j) \mid j = 1,\ldots,N+1\}$ be the points on the possibility distribution. Therefore, P_1 and P_2 can be calculated using the following expressions.

$$P_1 = \sum_{j=1}^{N} A_j \tag{A1}$$

$$P_2 = \frac{\sum_{j=1}^{N} A_j \left(\frac{X_{j+1}+X_j}{2}\right)}{\sum_{j=1}^{N} A_j} \tag{A2}$$

Here, the expression of A_j is as follows.

$$A_j = \frac{\left|\left(X_{j+1} - X_j\right)\right|\left(Y_{j+1} + Y_j\right)}{2} \tag{A3}$$

Let (u_X, w_X) and v_X be the core and support of TFN, respectively. Therefore, P_3 and P_4 can be calculated using the following expressions.

$$P_3 = v_X \tag{A4}$$

$$P_4 = w_X - u_X \tag{A5}$$

Let $P_{i(\text{real})}$ denote the values of P_i, $i = 1,\ldots,4$, for the possibility distribution and TFN corresponding to the real signal. Let $P_{i(\text{simulated})}$ denote the values of P_i, $i = 1,\ldots,4$, for the possibility distribution and TFN corresponding to the simulated signal. This results in a measure of error denoted as e_i. This error can be calculated as follows.

$$e_i = \left|\frac{P_{i(\text{simulated})} - P_{i(\text{real})}}{P_{i(\text{real})}}\right| \tag{A6}$$

The effect of e_j can be aggregated by calculating the average error denoted as Error (E). Thus, the following expression holds.

$$E = \frac{\sum_{i=1}^{4} e_i}{4} \tag{A7}$$

The simulated signal exhibiting the minimal E matches the real signal as closely as possible. Therefore, E can be used in monitoring a manufacturing process. This is implemented in the proposed DT.

References

[1] L.Da Xu, E.L. Xu, L. Li, Industry 4.0: state of the art and future trends, Int. J. Prod. Res. 56 (2018) 2941–2962, https://doi.org/10.1080/00207543.2018.1444806.

[2] A. Kusiak, Smart manufacturing, Int. J. Prod. Res. 56 (2018) 508–517, https://doi.org/10.1080/00207543.2017.1351644.

[3] G. Schuh, R. Anderl, J. Gausemeier, M. ten Hompel, W. Wahlster (Eds.), Industry 4.0 Maturity Index. Managing the Digital Transformation of Companies, Acatech STUDY, Herbert Utz Verlag, Munich, 2017.

[4] J. Zhou, P. Li, Y. Zhou, B. Wang, J. Zang, L. Meng, Toward new-generation intelligent manufacturing, Engineering 4 (2018) 11–20, https://doi.org/10.1016/j.eng.2018.01.002.

[5] G. Morteza, The future of manufacturing industry: a strategic roadmap toward Industry 4.0, J. Manuf. Technol. Manag. 29 (2018) 910–936, https://doi.org/10.1108/JMTM-02-2018-0057.

[6] D.A. Rossit, F. Tohmé, M. Frutos, A data-driven scheduling approach to smart manufacturing, J. Ind. Inf. Integr. 15 (2019) 69–79, https://doi.org/10.1016/j.jii.2019.04.003.

[7] Y. Lu, Industry 4.0: a survey on technologies, applications and open research issues, J. Ind. Inf. Integr. 6 (2017) 1–10, https://doi.org/10.1016/j.jii.2017.04.005.

[8] E. Oztemel, S. Gursev, Literature review of Industry 4.0 and related technologies, J. Intell. Manuf. 31 (2020) 127–182, https://doi.org/10.1007/s10845-018-1433-8.

[9] L. Monostori, B. Kádár, T. Bauernhansl, S. Kondoh, S. Kumara, G. Reinhart, O. Sauer, G. Schuh, W. Sihn, K. Ueda, Cyber-physical systems in manufacturing, CIRP Ann. 65 (2016) 621–641, https://doi.org/10.1016/j.cirp.2016.06.005.

[10] X. Yao, J. Zhou, Y. Lin, Y. Li, H. Yu, Y. Liu, Smart manufacturing based on cyber-physical systems and beyond, J. Intell. Manuf. 30 (2019) 2805–2817, https://doi.org/10.1007/s10845-017-1384-5.

[11] J. Lee, B. Bagheri, H.A. Kao, A cyber-physical systems architecture for industry 4.0-based manufacturing systems, Manuf. Lett. 3 (2015) 18–23, https://doi.org/10.1016/j.mfglet.2014.12.001.

[12] Y. Lu, J. Cecil, An internet of things (IoT)-based collaborative framework for advanced manufacturing, Int. J. Adv. Manuf. Technol. 84 (2015) 1141–1152, https://doi.org/10.1007/s00170-015-7772-0.

[13] J. Pang, N. Zhang, Q. Xiao, F. Qi, X. Xue, A New Intelligent and Data-Driven Product Quality Control System of Industrial Valve Manufacturing Process in CPS, Comput. Commun. 175 (2021) 25–34.

[14] R.S. Nakayama, M. de Mesquita Spínola, J.R. Silva, Towards I4.0: a comprehensive analysis of evolution from I3.0, Comput. Ind. Eng. 144 (2020), 106453, https://doi.org/10.1016/j.cie.2020.106453.

[15] J. Morgan, M. Halton, Y. Qiao, J.G. Breslin, Industry 4.0 smart reconfigurable manufacturing machines, J. Manuf. Syst 59 (2021) 481–506, https://doi.org/10.1016/j.jmsy.2021.03.001.

[16] A.K. Ghosh, A.M.M.S. Ullah, A. Kubo, T. Akamatsu, D.M. D'Addona, Machining phenomenon twin construction for industry 4.0: a case of surface roughness, J. Manuf. Mater. Process. 4 (2020), https://doi.org/10.3390/jmmp4010011.

[17] S. Aheleroff, X. Xu, R.Y. Zhong, Y. Lu, Digital twin as a service (DTaaS) in industry 4.0: an architecture reference model, Adv. Eng. Inform. 47 (2021), 101225, https://doi.org/10.1016/j.aei.2020.101225.

[18] F. Tao, Q. Qi, L. Wang, A.Y.C. Nee, Digital twins and cyber–physical systems toward smart manufacturing and industry 4.0: correlation and comparison, Engineering 5 (2019) 653–661, https://doi.org/10.1016/j.eng.2019.01.014.

[19] R. Minerva, G.M. Lee, N. Crespi, Digital twin in the IoT Context: a survey on technical features, scenarios, and architectural models, Proc. IEEE 108 (2020) 1785–1824, https://doi.org/10.1109/JPROC.2020.2998530.

[20] A. Fuller, Z. Fan, C. Day, C. Barlow, Digital Twin, Enabling technologies, challenges and open research, IEEE Access 8 (2020) 108952–108971, https://doi.org/10.1109/ACCESS.2020.2998358.

[21] M.J. Kaur, V.P. Mishra, P. Maheshwari, in: M. Farsi, A. Daneshkhah, A. Hosseinian-Far, H. Jahankhani (Eds.), The Convergence of Digital Twin, IoT, and Machine Learning: transforming Data into Action, Springer International Publishing, Cham, 2020, pp. 3–17, https://doi.org/10.1007/978-3-030-18732-3_1.

[22] Z. Jiang, Y. Guo, Z. Wang, Digital twin to improve the virtual-real integration of industrial IoT, J. Ind. Inf. Integr. 22 (2021), 100196, https://doi.org/10.1016/j.jii.2020.100196.

[23] IIC, Digital twins for industrial applications: definition, business values, design aspects, standards and use cases. (2020), https://www.iiconsortium.org/pdf/IIC_Digital_Twins_Industrial_Apps_White_Paper_2020-02-18.pdf (2020).

[24] F. Tao, J. Cheng, Q. Qi, M. Zhang, H. Zhang, F. Sui, Digital twin-driven product design, manufacturing and service with big data, Int. J. Adv. Manuf. Technol. 94 (2018) 3563–3576, https://doi.org/10.1007/s00170-017-0233-1.

[25] T. Ruppert, J. Abonyi, Integration of real-time locating systems into digital twins, J. Ind. Inf. Integr. 20 (2020), 100174, https://doi.org/10.1016/j.jii.2020.100174.

[26] S. Aheleroff, R.Y. Zhong, X. Xu, A digital twin reference for mass personalization in industry 4.0, Procedia CIRP 93 (2020) 228–233, https://doi.org/10.1016/j.procir.2020.04.023.

[27] J. Chen, P. Hu, H. Zhou, J. Yang, J. Xie, Y. Jiang, Z. Gao, C. Zhang, Toward intelligent machine tool, Engineering 5 (2019) 679–690, https://doi.org/10.1016/j.eng.2019.07.018.

[28] W. Luo, T. Hu, Y. Ye, C. Zhang, Y. Wei, A hybrid predictive maintenance approach for CNC machine tool driven by Digital Twin, Robot. Comput. Integr. Manuf. 65 (2020), 101974, https://doi.org/10.1016/j.rcim.2020.101974.

[29] X. Tong, Q. Liu, S. Pi, Y. Xiao, Real-time machining data application and service based on IMT digital twin, J. Intell. Manuf. 31 (2020) 1113–1132, https://doi.org/10.1007/s10845-019-01500-0.
[30] R.H. Jhaveri, R. Tan, A. Easwaran, S.V. Ramani, Managing industrial communication delays with software-defined networking, in: Proceedings of the 25th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA) IEEE, 2019, pp. 1–11, https://doi.org/10.1109/RTCSA.2019.8864557.
[31] P. Ferrari, A. Flammini, E. Sisinni, S. Rinaldi, D. Brandão, M.S. Rocha, Delay estimation of industrial IoT applications based on messaging protocols, IEEE Trans. Instrum. Meas. 67 (2018) 2188–2199, https://doi.org/10.1109/TIM.2018.2813798.
[32] E. Bradley, H. Kantz, Nonlinear time-series analysis revisited, Chaos Interdiscip. J. Nonlinear Sci. (2015) 25, https://doi.org/10.1063/1.4917289.
[33] AMMS Ullah, Modeling and simulation of complex manufacturing phenomena using sensor signals from the perspective of Industry 4.0, Adv. Eng. Informatics 39 (2019) 1–13, https://doi.org/10.1016/j.aei.2018.11.003.
[34] R. Teti, T. Segreto, A. Caggiano, L. Nele, Smart multi-sensor monitoring in drilling of CFRP/CFRP composite material stacks for aerospace assembly applications, Appl. Sci. 10 (2020) 758, https://doi.org/10.3390/app10030758.
[35] T.P. Raptis, A. Passarella, M. Conti, Data management in industry 4.0: state of the art and open challenges, IEEE Access 7 (2019) 97052–97093, https://doi.org/10.1109/ACCESS.2019.2929296.
[36] S. Zoppi, A. Van Bemten, H.M. Gürsu, M. Vilgelm, J. Guck, W. Kellerer, Achieving hybrid wired/wireless industrial networks with WDetServ: reliability-based scheduling for delay guarantees, IEEE Trans. Ind. Inform. 14 (2018) 2307–2319, https://doi.org/10.1109/TII.2018.2803122.
[37] H. Mo, W. Wang, M. Xie, J. Xiong, Modeling and analysis of the reliability of digital networked control systems considering networked degradations, IEEE Trans. Autom. Sci. Eng. 14 (2017) 1491–1503, https://doi.org/10.1109/TASE.2015.2443132.
[38] H. Zhang, Y. Shi, J. Wang, H. Chen, A new delay-compensation scheme for networked control systems in controller area networks, IEEE Trans. Ind. Electron. 65 (2018) 7239–7247, https://doi.org/10.1109/TIE.2018.2795574.
[39] J.W. Guck, M. Reisslein, W. Kellerer, Function split between delay-constrained routing and resource allocation for centrally managed QoS in industrial networks, IEEE Trans. Ind. Inform. 12 (2016) 2050–2061, https://doi.org/10.1109/TII.2016.2592481.
[40] M. Yagi, Y. Sawada, State estimation for networked system with random delay using Kalman filter, Proc. ISCIE Int. Symp. Stoch. Syst. Theory Appl. (2014) 9–14, https://doi.org/10.5687/sss.2014.9.
[41] Y. Fan, S. Liu, X. Liu, A new predictive control systems with network delays in the feedback and forward channels, in: Proceedings of the Chinese Control Decision Conference, 2011, pp. 912–916, https://doi.org/10.1109/CCDC.2011.5968313.
[42] J. Wu, L. Zhang, T. Chen, Model predictive control for networked control systems, Int. J. Robust Nonlinear Control. 19 (2009) 1016–1035, https://doi.org/10.1002/rnc.1361.
[43] Y. Sun, Y. Huo, Robust static output feedback control for networked control system with random time delay, in: Proceedings of the 2nd International Conference on Advanced Computer Control, ICACC, 2010, pp. 547–551, https://doi.org/10.1109/ICACC.2010.5487125.
[44] L. Guo, H. Gu, Robust stability of discrete systems with uncertainties and random delay, in: Proceedings of the International Conference on Measuring Technology and Mechatronics Automation, ICMTMA, 2010, pp. 291–294, https://doi.org/10.1109/ICMTMA.2010.559.
[45] J. Baillieul, P.J. Antsaklis, Control and communication challenges in networked real-time systems, Proc. IEEE 95 (2007) 9–28, https://doi.org/10.1109/JPROC.2006.887290.
[46] E. Bijami, M.M. Farsangi, A distributed control framework and delay-dependent stability analysis for large-scale networked control systems with non-ideal communication network, Trans. Inst. Meas. Control 41 (2018) 768–779, https://doi.org/10.1177/0142331218770493.
[47] C. Zunino, A. Valenzano, R. Obermaisser, S. Petersen, Factory communications at the dawn of the fourth industrial revolution, Comput. Stand. Interfaces 71 (2020), 103433, https://doi.org/10.1016/j.csi.2020.103433.
[48] S. Kontogiannis, G. Kokkonis, Proposed fuzzy real-time haptics protocol carrying haptic data and multisensory streams, Int. J. Comput. Commun. Control (2020) 15, https://doi.org/10.15837/ijccc.2020.4.3842.
[49] C. Xia, X. Jin, C. Xu, Y. Wang, P. Zeng, Real-time scheduling under heterogeneous routing for industrial internet of things, Comput. Electr. Eng. 86 (2020), 106740, https://doi.org/10.1016/j.compeleceng.2020.106740.
[50] R. Basir, S. Qaisar, M. Ali, M. Aldwairi, M.I. Ashraf, A. Mahmood, M. Gidlund, Fog computing enabling industrial internet of things: state-of-the-art and research challenges, Sensors 19 (2019), https://doi.org/10.3390/s19214807.
[51] F. Wang, S. Shu, F. Lin, Robust networked control of discrete event systems, IEEE Trans. Autom. Sci. Eng. 13 (2016) 1528–1540, https://doi.org/10.1109/TASE.2016.2588527.
[52] A.S. Ullah, Fundamental issues of concept mapping relevant to discipline-based education: a perspective of manufacturing engineering, Educ. Sci. 9 (2019) 228, https://doi.org/10.3390/educsci9030228.
[53] A.S. Ullah, Concept map and knowledge, Educ. Sci. 10 (2020) 246, https://doi.org/10.3390/educsci10090246.
[54] A.M.M. Sharif Ullah, M. Shamsuzzaman, Fuzzy Monte Carlo Simulation using point-cloud-based probability–possibility transformation, Simulation 89 (2013) 860–875, https://doi.org/10.1177/0037549713482174.
[55] A.M.M. Ullah, Surface roughness modeling using Q-sequence, Math. Comput. Appl. 22 (2017) 33, https://doi.org/10.3390/mca22020033.
[56] A.M.M.S. Ullah, D. D'Addona, N. Arai, DNA based computing for understanding complex shapes, Biosystems 117 (2014) 40–53, https://doi.org/10.1016/j.biosystems.2014.01.003.
[57] S. Imran Shafiq, E. Szczerbicki, C. Sanin, Decisional-DNA based smart production performance analysis model, Cybern. Syst. 50 (2019) 154–164, https://doi.org/10.1080/01969722.2019.1565122.
[58] T. Berners-Lee, J. Hendler, O. Lassila, The semantic web, Sci. Am. 284 (2001) 34–43.
[59] B.H. Prasetio, H. Tamura, K. Tanno, Deep time-delay Markov network for prediction and modeling the stress and emotions state transition, Sci. Rep. 10 (2020) 18071, https://doi.org/10.1038/s41598-020-75155-w.
[60] M. Salins, K. Spiliopoulos, Markov processes with spatial delay: path space characterization, occupation time and properties, Stoch. Dyn. 17 (2017), 1750042, https://doi.org/10.1142/S0219493717500423.
[61] A.K. Ghosh, A.M.M.S. Ullah, A. Kubo, Hidden Markov model-based digital twin construction for futuristic manufacturing systems, Artif. Intell. Eng. Des. Anal. Manuf. 33 (2019) 317–331, https://doi.org/10.1017/S089006041900012X.