Use of Digital Twins and Digital Threads for Subway Infrastructure Monitoring

Alexander Vodyaho[1], Elena Stankova[2], Nataly Zhukova[3], Alexey Subbotin[1](✉), and Michael Chervontsev[1]

[1] Saint-Petersburg State Electrotechnical University, Professor Popov Street 5, 197376 St. Petersburg, Russia
alesu1543@gmail.com
[2] St. Petersburg State University, 7-9, Universitetskaya nab., 199034 St. Petersburg, Russia
[3] St. Petersburg Institute of Informatics and Automation of the Federal Research Center at the Russian Academy of Sciences (FRC RAS), 14th Line V.O. 39, 199178 St. Petersburg, Russia

Abstract. The article deals with the problem of using digital twins and digital threads in complex distributed cyber-physical systems with a high level of structural and functional dynamics. A three-level model of the life cycle of a complex cyber-physical system is proposed. At the upper level, the observed system is described in terms of a continuous architecture. At the middle level, the observed system is described in terms of an agile architecture. At the lower level, the observed system is described in terms of a multigraph, which allows describe both the observed system structure and behavior. In the case study the solution of the problem of monitoring the state of the subway infrastructure is considered. The proposed approach has shown its effectiveness and can be applied in other domains such as smart cities.

Keywords: Digital twin · Digital thread · Event monitoring and visualization · Smart city · Machine learning · Neural network algorithms

1 Introduction

One of the distinguishing features of the modern stage of society's development is very rapid changes in people's lives and production methods. Modern companies and organizations are transforming their strategies, business models, products and services, as well as business processes and information systems by increasing the level of their digitalization [1] with the help of intelligent services and products usage. The potential of the Internet and related digital technologies, such as the Internet of Things, artificial intelligence, data analysis, computing services, cloud computing, mobile systems, collaboration networks and cyber-physical systems (CPS), are strategic factors that allow the development of digital platforms and ecosystems [2] based on intelligent services.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2022
O. Gervasi et al. (Eds.): ICCSA 2022 Workshops, LNCS 13380, pp. 693–707, 2022.
https://doi.org/10.1007/978-3-031-10542-5_48

Currently, digital transformation is the main trend of the modern business development. Digital technologies are the main driving forces of digitalization, as they are a way of doing business. Digital technologies provide three main opportunities for a changing business [1]: i) ubiquitous data distribution; ii) unlimited connectivity; iii) huge computing power.

Using the term "digitalization" [6], we mean something more than just digital technologies. Digitalization [3] is the result of digital transformation [4, 5], which is a transition from analog-digital to fully digital technology. Thanks to this replacement, analog media are replaced by digital media with the same commercial value, while the digital addition functionally enriches the corresponding converted analog media.

Thus, digitalization [6] is more associated with the transition of processes to highly automated digital business processes, and not only with communication via the Internet.

It is obvious that digital transformations [4, 6] are closely related to digital technologies.

Digitalization [3] promotes the development of large-scale IT systems with distributed structures, such as the Internet of Things or mobile systems. Data, Information and Knowledge (DIK) are fundamental concepts of our daily activities and are the driving force of digital transformation [4] of modern global society [3].

The implementation of the digital transformation of society and industry requires reaching a new level of complexity of the created man-made systems. As a measure of the complexity of the created man-made systems can serve not only the number of elements, the number of hierarchy levels, the number of connections between the elements of the system, but also the complexity of the behavior of systems, which can be expressed in the expansion of the set of implemented functions, in the complexity of the functions themselves and the ability of systems to adaptation. New classes of systems are constantly appearing, in addition, the concept of a complex system is expanding. It includes such concepts as a complex phenomenon, a complex process, while the elements of a complex system are not only software components, but also elements of a very different nature (technological equipment, transport systems, natural phenomena, biological systems), as well as people responsible for the extraction, accumulation and use of knowledge. This approach is used in the construction of cyber-physical systems (CPS), the functioning of which can be described using models that are not directly tied to a specific implementation [1–11].

It can be argued that modern anthropogenic systems are mostly large-scale distributed heterogeneous systems that can be classified as System of Systems (SoS) with an agile architecture [12].

The emergence of fundamentally new classes of complex systems and large scale technical complexes requires the use of new architectural solutions and design approaches. When building modern information-oriented systems, such new architectural paradigms as flexible architectures, continuous architectures, CPS, and ambient intelligence systems are increasingly used. Platforms such as cloud and fog computing, the Internet of Things, and the Industrial Internet of Things are actively used as platforms.

Model-based approaches, which are widely used in system design, can be considered as such approaches. In particular, in recent years the Digital Twins (DT) and Digital Threads (DTh) paradigms have been widely used for SoS building.

DT and DTh paradigms can be considered both as implementations of a model-based approach and as virtualization mechanisms. DT can be used at different stages of the system Life Cycle (LC), in particular at the stages of production and operation. The idea of DTh can be viewed as a generalization of the idea of DT, according to which digital models (twins) accompany the system throughout its LC.

The use of DT can be seen as a means of integrating heterogeneous elements into a system, and the use of DTh can be conceded as an approach to the efficient organization of the production process.

The key problem that limits the widespread use of both DT and DTh is the rather high cost of developing and maintaining models up to date. In this context, the task of automating the process of building models comes to the fore, especially when it comes to systems with variable structure and behavior, when building common models is impossible.

2 Digital Twins and Digital Threads

There are a number of definitions of the concept of DT [13]. The most general definition of DT according to ISO/IEC is the following "*a digital twin is a virtual representation of real-world entities and processes, synchronized at a specified frequency and fidelity*".

One can define the following typical approaches to the use of DT: i) DT is a controller of a physical entity; ii) DT is an intelligent controller of a physical entity; iii) DT is a simplified model of the Observed System (ObS); iv) DT is a full functional analog of the ObS; v) DT is the access port to other systems, in particular, physical systems.

In the first case, DT is conceded as a passive adapter which can track the status and synchronize with a physical entity. In the second case, the DT can support ObS agility mechanisms. In the third case, DT is a simplified ObS models which are built for the needs of the concrete stakeholder. In the fourth case, DT is considered as a full functional ObS analog. In this case ObS can be replaced by the DT. In the fifth case, DT is considered as an instrument of SoS building. Each system that is a part of a SoS can have an arbitrary number of DTs that describe the system in the vocabularies that are understandable by other SoS elements. In this case, DT can be considered as a service available to other SoS elements.

A DTh can be defined as a set of models belonging to different stages of the LC and a set of mechanisms for the transformation of these models. It should be noted that the models at different stages of the LC differ significantly, and the task of transforming the models is quite complex [13].

It can be argued that nowadays the concept of DT is a multidimensional concept and is most closely related and based on paradigms such as evolutionary, continuous and agile architecture [9].

When using approaches based on DTh, the task of transforming models related to different stages of the system LC becomes important.

This article discusses the possibility of using the DT paradigm in relation to large and complex systems with a high level of structural and functional dynamics. Special attention is paid to the stage of functioning where the data collection (acquisition) task about the current state of ObS is of high importance.

3 Three-Level Model of a Continuous Agile Architecture

The large-scale CPS can be described with the help of a three-level model. The proposed model describes the LC in terms of a continuous agile architecture in five life cycle stages and three levels.

The structure of the model is shown in Fig. 1.

Top level: Continuous architecture
Intermediate: Agile architecture
Lower level: Structural-functional run-time model

Fig. 1. Continuous agile architecture model.

At the top level, the ObS is described in terms of a continuous architecture LC.

At the middle level, the system architecture as an agile architecture is described.

At the lower level, structural and functional models are used that describe the functioning of the ObS in terms of current states.

Upper Level. At this level, the process of transforming the architectural representations of the system during the transition between different stages of the LC is described (Fig. 2).

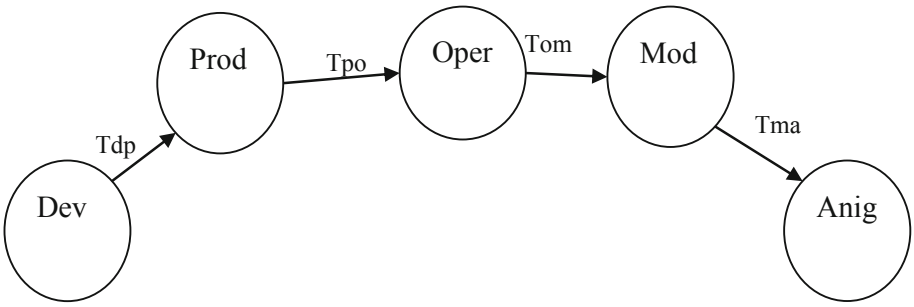

Fig. 2. Artifact transformation.

In general, 5 phases of the LC cycle can be distinguished: Development (Dev), Production (Prod), Operation (Oper), Modernization (Mod), Anigilation (Anig). The chain of transformations in terms of artifacts (Art) can be represented as follows: Art Dev → Tdp → Art Prod → Tpo → Art Oper → Tom → Art Mod → Tma → Art Anig where Tdp, Tpo, Tom, Tma are the operations of artifacts transformations between LC phases.

At each phase of the LC, a certain set of artifacts is formed. Artifacts belong to the classes defined as Art::=<Model|Code|Physical entity>. Each phase uses its own set of artifacts.

There are 4 basic types of transformations that can be defined:

$M_i \rightarrow M_{i-1}$, M → Code, M → PhE where $M_i \rightarrow M_{i-1}$ is a transformation of a model of the i-th level to the models of the i−1 level, PhE is a physical entity.

In addition, more complex transformations can be distinguished (Fig. 3), such as operation of adding extra models, e.g. additional views (Fig. 3,a) and operation of DT creation (Fig. 3,b). Other types of transformations are also possible.

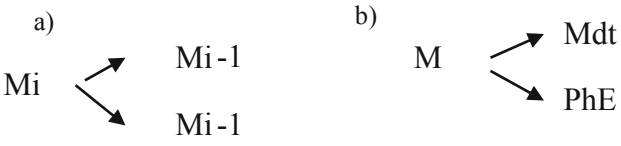

Fig. 3. Compound transformations.

In the general case, at each LC stage its own set of entities is used.

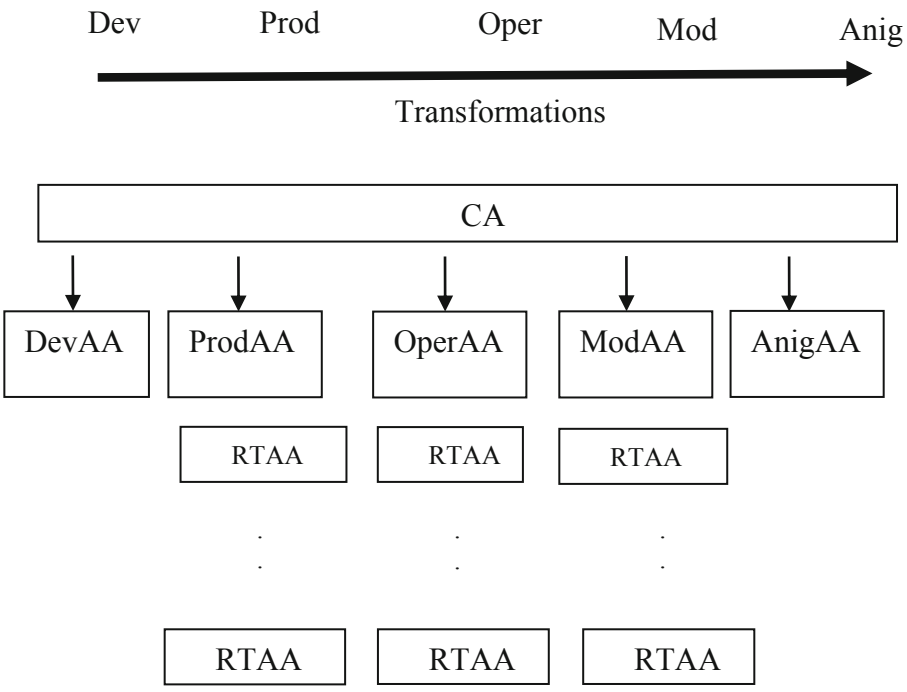

Fig. 4. Artifact system structures.

The structure of the artifact system is shown in Fig. 4. At the top level, one can talk about the single model. At the middle level, each phase uses its own artifacts. Each phase has its own Agile Architecture (AA). To each AA architecture a set of Run-Time Agile Architectures (RTAA) is assigned.

Each phase uses its own artifact systems.

Models can be divided into two groups: models that describe classes and models that describe entity instances.

A model and a model entity can be in the following relationship: i) implementation of the M → E model; ii) building the model E → M; iii) synchronization of a model and an entity instance M ↔ E.

In Fig. 5 an automaton model is presented that describes the behavior of the ObS in terms of the change of architectural states (ASt) under the action of external or internal events (Ev) and control actions (RecScr), which transfer the ObS to a new architectural state. Using this model, it is possible to describe the agile architecture [12–16, 21] in terms of changes in architectural states.

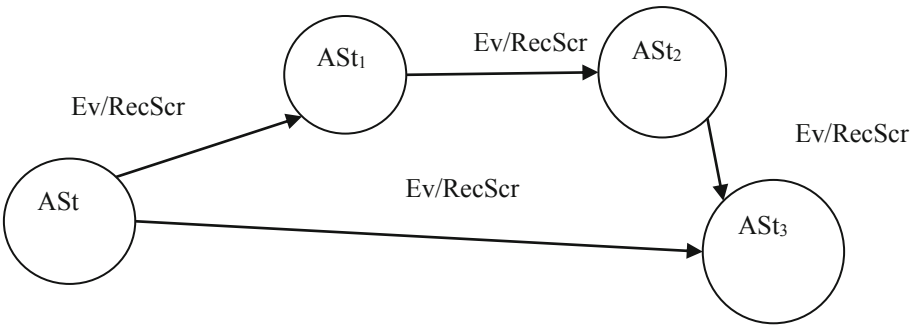

Fig. 5. High level automaton model.

The architectural state ASt can be defined as ASt = <AL, R2LLM>, where AL is an attribute list and R2LLM is the reference to LLM. The automaton presented in Fig. 4 is a distributed Relatively Finite State Operating Automaton (MLRFSOA) operating in discrete space and discrete time, which describes the level-by-level behavior of the ObS.

MLRFSOA is an automaton, in which the functions of transitions and outputs explicitly depend on time, Ai + 1 = F (Ai, t) or A = (X(t), Y(t), S(t), T (t), s0), where Xt are input signals, Yt are outputs, St are internal states, Tt is a transition function, s0 is an initial state. Since the automaton operates in discrete time and with discrete states, it can be assumed that at each moment of time the automaton is in the state which can be defined as A = (X(si), Y(si), S(si), T(si), s0). It is known that automata with a variable structure can generally be reduced to an ordinary automaton, but the number of the states becomes very large, which makes it difficult to define them. This can be dealt with in 2 ways: by generating a new automaton "on the fly" or by considering special cases when the number of new states is a countable set. It seems appropriate to consider special cases, in particular, when an automaton switches to a new state, only separate elements of its description are changed. In this case, an automaton with a variable structure can

be defined as a set of automata, $Ai \in A$, Ai = Ai − 1, Δi − 1, i, where Δ is a set of actions to restructure the automaton, the execution of which leads to the construction of the required automaton.

It should be noted that only in the simplest cases, the structure of the considered automaton can be determined at the design stage. Most often, especially when ObS belongs to the SoS class, knowledge about the structure of the automaton is incomplete. In this case, it is necessary to solve the problem of constructing (synthesizing) an automaton. These issues are considered in sufficient detail in the works of the authors [17–20].

LLM describes a specific ASt in terms of structure and behavior and is considered as a Structural and Functional Model (SFM), which can be defined as LLM = <MS, MB>, where MS is a structural model, and MB is a behavior model. It is often necessary to obtain information only about the structure or behavior, or even about certain aspects of the structure and/or behavior, for example, about the relationships between elements.

4 Structural-Functional Run-Time Model

Suggested SFM describes the run time structure and behavior of the ObS.

The following basic requirements for the structural-functional runtime models can be formulated.

The main requirements for structural models are the following: i) the ability to describe a constantly changing multi-level structure of arbitrary complexity; ii) the ability to automatically build a structural model based on limited information received in the form of logs.

The main requirements for behavioral models are the following: i) the ability to track the progress and correctness of Business Processes (BP) execution by incoming logs, ii) the ability to adjust the structure of BP when their structure changes.

An additional requirement for both models is the relative ease of implementation.

Because of it the most preferable is to use graph models, e. g. work flow graph-based models, there are a number of effective algorithms and tools for work flow graph processing such as process mining algorithms.

The SFM can be presented as a bipartite multigraph constructed on the basis of the workflow graph: SFM = <OPV, OPA, RV, RA, ORA>, where OPV are operator vertices, OPA (OP arcs) are arcs connecting operator vertices, RV are resource vertices, RA (R arcs) are connections between resource vertices, ORA (OR arcs) are connections between operator vertices and resource vertices.

OPV are operators of any level of complexity, operators are connected by OP arcs, which can be colored in one of 4 colors: arcs through which data is transmitted, arcs through which control signals are transmitted that allow OPV execution, arcs through which requests for operator execution are transmitted, arcs through which signals about the availability of resources are distributed. There are 16 different ways to check the readiness of operators for execution, which determine the strategies for managing the Business Process (BP). BP can be described with the help of a modified workflow graphs. Modified process mining algorithms can be used to construct data flow, control flow, and demand flow graphs [21]. To build a resource graph, which is essentially a structural graph, different mechanisms can be used at different levels.

The dynamics of the execution of the SFM can be described in terms of calculating the readiness of the operator to execute. The following conditions act as conditions for the operator's readiness: the availability of initial data, the availability of the necessary resources, and the availability of a request and the presence of a control signal to launch the operator.

The decision on the readiness of the operator is made in dynamics. However, some of the readiness conditions can be checked statically at the level of input language schemas or even at the algorithm level. In other words, a limited number of conditions are checked in dynamics, the rest are considered to be fulfilled by default.

5 Data Collection Virtual Machine

The conceptual structure of the virtual machine for supporting data collection is shown in Fig. 6.

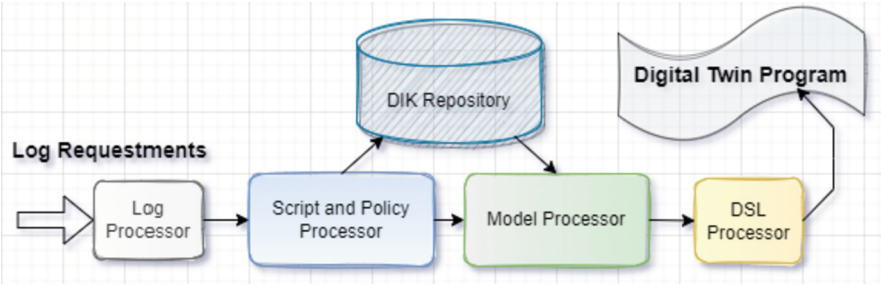

Fig. 6. Virtual machine structure.

The virtual machine consists of 4 asynchronously functioning virtual machines (processors): a log processor (Log P), a model processor (MP), a script and policy processor (SPP), a Domain Specific Language (DSL) processor (DSLP) and a DIK repository (DIKR). All listed elements are distributed.

Log P is responsible for logs processing through the implementation of scripts. MP implements two main functions: maintaining the adequacy of models and generating responses to user requests. The Script and policy processor is responsible for generating policy-based scripts. DSL Pr is an interpreter of a set of DSLs with which different categories of users work.

Algorithm for keeping the model up to date.

1. Launching the monitoring procedure.
2. Reception of logs.
3. Cleaning and sorting logs.
4. If correction of the model is required, then continue, otherwise go to step 2.
5. Correction of the model and transition to step 2.

Algorithm for processing user requests.

1. Waiting for a request from the user.
2. Receiving a request and transforming DSL → MQL (Model Query Language).
3. Request to the model in MQL.
4. If the answer is not received then continue, otherwise MQL → DSL and go to step 1.
5. Policy definition.
6. Synthesis of the script.
7. Execution of the script.
8. Construction of the required ObS model.
9. Request for a new model.
10. MQL → DSL and transition to item 1.

6 Case Study

The usage of the suggested approach can be illustrated on the example of a subway station, which can be conceded as an element of smart city ecosystem [1, 4, 5].

The functional model for the DT of a subway station is shown in Fig. 7. The operators are presented in ovals, the arcs that connect the vertices, define the conditions for their execution. The model assumes that first the data is received, then its type is determined. Depending on the type of data (video, audio, sensors), further information processing takes place. The video is divided into frames, then a series of frames is processed in a fog computing environment or cloud using machine learning algorithms for identifying objects on the images. The audio is also divided into frames and processed by machine learning algorithms.

Sensors are initially defined as either analog or discrete. If the sensors are discrete, then the events that occur on the subway stations are determined immediately, and if the signal from the sensors is analog, then the event is determined using machine learning. Processing the signal from an analog sensor resembles audio processing, which differs only in amplitude and certain sine wave characteristics. In any case, regardless of the type of information (video, audio, sensors), data and results of its processing are transferred to the database and used to build digital twins of the metro stations.

The algorithm for building DT of the metro stations contains the following steps:

1. Determining the type of the received data (video, audio, sensors).
2. If this is a video stream, then division into frames (with a frequency of 10 frames per second, but not more than 24 and a resolution from 720 × 1280 to 1920 × 1080 using a Panasonic Full HD 1080p HC-V730 camcorder, zoom 20×/60×).
3. If this is data from a sensor, then determine the type of the sensor (discrete or analog).
4. If this is audio stream, then the division into frames (based on the Meridian Lossless Packing (MLP) codec with a bit rate of 18 Mbit/s, divided into fragments from 2 to 5 s).
5. Determination of objects based on series of video frames using machine learning (using the methods of convolutional neural networks, deep learning, recurrent neural networks using the library TensorFlow 2.0 on Python 3.8).

6. Determination of events based on audio frames using machine learning (using deep learning neural network methods from frameworks PyTorch, CNTK and MXNet).
7. Determination of events based on data from sensors (using internal sensor algorithms that are based on the analysis of an analog signal, where the sinusoid crosses a certain set threshold with a certain intensity, which indicates a certain event).
8. Creation of logs.
9. Saving data and results of its processing to the database to build a DT.

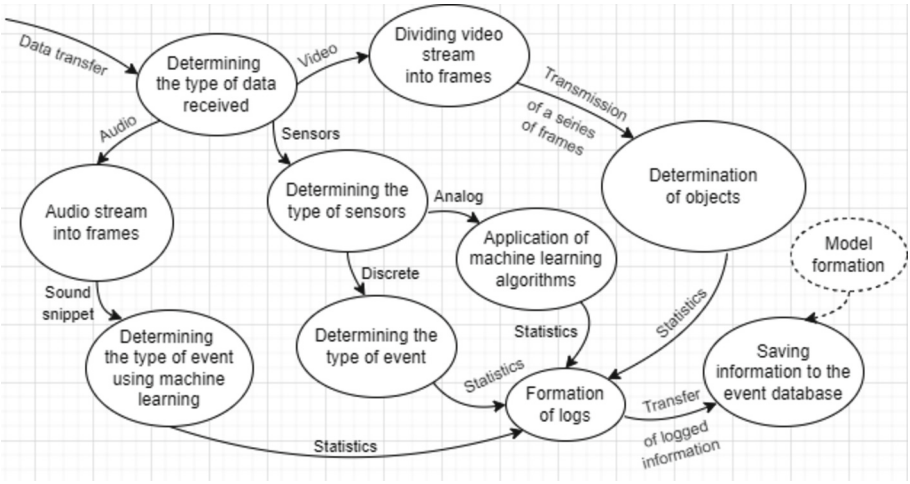

Fig. 7. Functional model of the DT of the subway station.

The functional model of the of the subway station monitoring is shown in Fig. 8. It starts from receiving data from the database, including the data about the objects of the subway station (structures, stairs, tracks, police offices, cash desks, etc.). Next, information is obtained from the event database. The type of event is determined and, depending on the degree of its danger and specificity, it is forwarded to the appropriate service. After contacting the service, logs are generated and information about the measures taken is stored. If a hardware error occurs, the information is stored in the database and the digital twin model of the station is re-built, taking into account the faulty hardware (cameras, sounds, sensors). If no dangerous events are identified, then the process is waiting for new information about events.

The steps of the algorithm of the subway station monitoring are the following:

1. Receiving data from the database.
2. Building the initial digital twin, that contains data about all objects of the station (call to algorithm 1).
3. Getting information about events.
4. Determining of the degree of danger of the event (the level of danger is determined on the basis of three sources of information: sensors, images and sounds).

5. If no dangerous events are identified, then the process is in the pending state, return to step 4.
6. If the event is identified as dangerous, then information about the event is transmitted to the appropriate service.
7. Formation of logs and reporting.
8. Saving to the database the information about the event for further analysis and the information about the hardware failures.

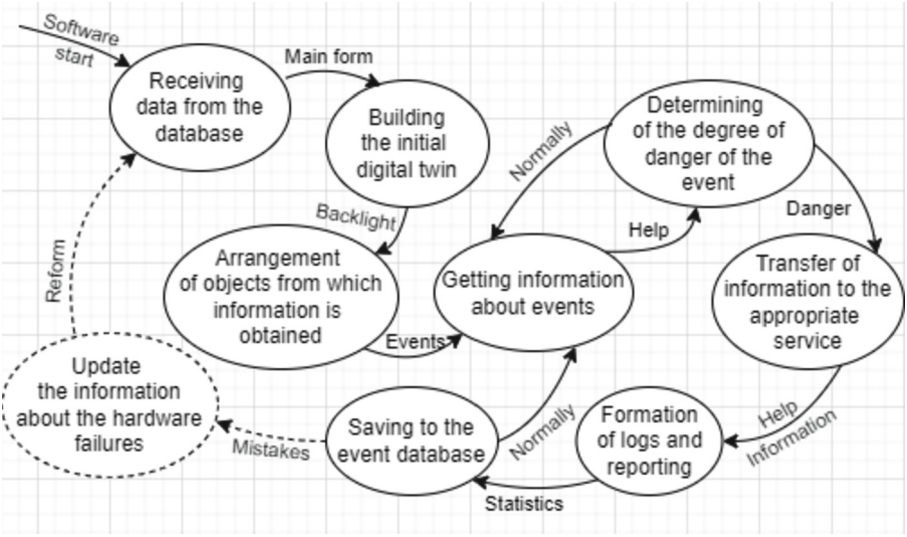

Fig. 8. Functional model of the of the subway station monitoring.

For dynamical building of the DTs of the subway stations a program was implemented. Due to using DT it provides for the employees of the subway the possibilities to determine the dangerous events quickly and thus prevent the incidents or take timely measures to help passengers. Usage of DT also allowed increase the accuracy of the event identification as for building the DT three data sources (cameras, sounds, sensors) are used and data is processed using machine learning algorithms [23, 26, 27]. Decision-making speed is enhanced because the employee works with an exact digital copy of the station that allows determine the place where the dangerous event has occurred accurately. The usage of DT also allowed increase the speed of reaction for the employees of the subway. The passengers can use the program to define location of subway employees when they need help.

The main frame of the developed program displays the objects of the station (stairs, exits, railway tracks, office premises, help kiosks, police departments, etc.), their status and the location of subway employees (Fig. 9). Also, the following information is available: processing time for a series of frames, connections, settings for encrypted channel, framing, finding images, rebuilding DTs and other. Also, it is possible to gather application performance metrics (accuracy, speed). At the bottom there are control buttons for

collecting data (Stop, Restart, Start) to control the process of events identification. At the bottom there are two more buttons (Timer and Logs) to open additional frames for viewing logs and set timer settings to stop measuring the speed and accuracy of events identification. In Fig. 9 both forms are shown on the top of the main frame. At the far right of the main frame of the program there are buttons (with arrows) for moving around the entire platform, since the platform occupies a considerable area and is much larger than the size of the screen. At the bottom right are settings for collecting data (sources of three types: Cameras, Audio, Sensors) and settings for managing history files (viewing in an external editor by clicking the Open button and completely clearing the logs by clicking the Clear button). Additional settings are opened by clicking the Advanced button. By default, the program logs are stored in text files in the "/opt/mnt/tw/logs/" directory, which can be changed.

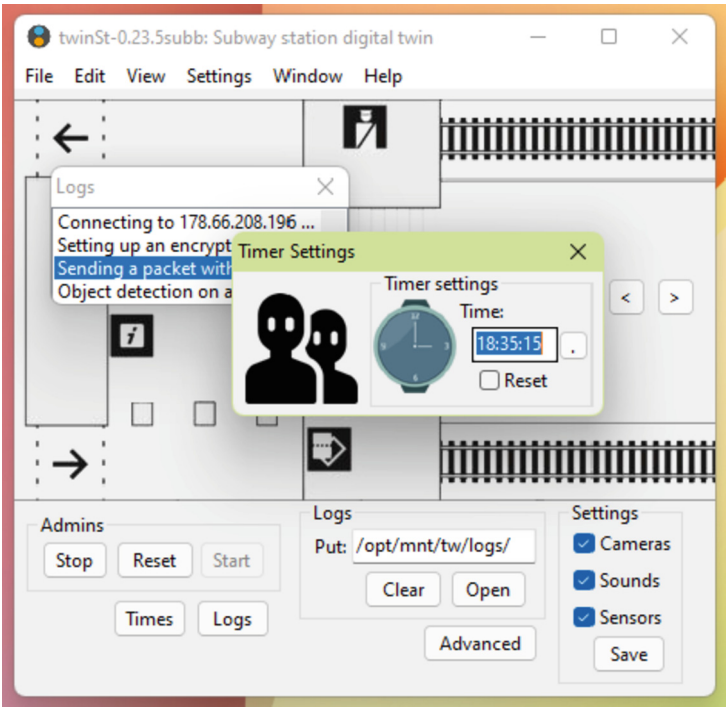

Fig. 9. The main frame of the program that implements a digital twin of a subway station.

The process of building and rebuilding the DT takes place in the cloud or fog computing environment. Python scripts that use TensorFlow for identifying events using machine learning are executed in the cloud. The log preprocessing is also located on the servers in the cloud.

After measuring the accuracy of the events identification, the following results were obtained (Table 1). The accuracy of event detection using the digital twin has increased by 10.56%.

Table 1. Event identification accuracy.

Event	Only logs are processed (visualization and monitoring program is not used)	Logs are processed and visualization and monitoring program is used	DT is used	Increase of the accuracy when using digital twin
Hanging on the edge of the platform	0.71	0.83	0.94	11.7%
Falling things and objects on the railroad tracks	0.78	0.86	0.97	11.3%
Running or fast moving on a platform	0.75	0.84	0.95	11.58%
Smoky or hazy environment	0.73	0.85	0.92	7.6%
Average				10.56%

To reproduce the experiments, it is necessary to develop a program for collecting data, a visualization program, and a "Digital Twin" program. Develop a program to summarize data or open logs in Microsoft Excel, apply filtration, summation, mathematical rounding and average. After that, fill in the resulting tables. The experiments were carried out on an HP Laptop 17-cp0092ur with technical specifications: 17.3″, 1600 × 900, SVA (TN + film), AMD 3020e, cores: 2 × 1.2 GHz, RAM 4 GB, SSD 256 GB, AMD Radeon Graphics, Windows 11. For cloud computing, an HP server was used at Immers (https://immers.cloud/) with the name "nvme.4.16.120" and technical characteristics: 16384 RAM, 4-core Intel Xeon Gold processor 6th Scalable series (Cascade Lake) using the OpenStack virtualization API with the "ninja_vm" command and pre-installed Debian 11.3 (https://www.debian.org/).

7 Conclusions

In the course of the study, a three-level model of a continuous agile architecture was developed. The use of this model allows describe all system LC stages.

The suggested approach was used for solving the problem of monitoring the state of the subway stations using three information sources (cameras, sounds and sensors). The research materials are recordings of video, sound and signals from sensors obtained at the subway station.

The program for DT building has been developed and measurements of its efficiency were carried out, which showed an increase in the accuracy of determining events by an average of 10.56%. The usage of DT approach has shown to be effective for solving the

problems of the subway, but it can be applied in more complex cases, in particular, at plants, factories, bridges and other complex structures.

References

1. Calinescu, R.C., Camara Moreno, J., Paterson, C.: Socio-cyber-physical systems: models, opportunities. In: Proceedings of the Open Challenges 5th International Workshop on Software Engineering for Smart Cyber-Physical Systems, Montreal, QC, Canada, 28 May 2019
2. Sanfelice, R.G.: Analysis and design of cyber-physical systems. A hybrid control systems approach. In: Rawat, D., Rodrigues, J., Stojmenovic, I. (eds.) Cyber-Physical Systems: From Theory to Practice. CRC Press, Boca Raton (2016). ISBN 978-1-4822-6333-6
3. Mahmood, Z.: Guide to Ambient Intelligence in the IoT Environment Principles, Technologies and Application, 289p. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-04173-1
4. Marques, G., Pitarma, R., MGarcia, N., Pombo, N.: Internet of Things architectures, technologies, applications, challenges, and future directions for enhanced living environments and healthcare systems: a review. Electronics **8**, 1081 (2019). https://doi.org/10.3390/electronics8101081
5. Korzun, D., Balandina, E., Kashevnik, A., Balandin, S., Viola, F.: Ambient Intelligence Services in IoT Environments: Emerging Research and Opportunities, 199p. IGI-Global, Hershey (2019). https://doi.org/10.4018/978-1-52258973-0
6. Wu, Y., Hu, F., Min, G., Zomaya, A.Y.: Big Data and Computational Intelligence in Networking, 530p. Taylor & Francis Group, LLC, Boca Raton (2018)
7. Weilkiens, T., Lamm, J., Roth, S., Walker, M.: Model-Based System Architecture, 375p. Wiley, Hoboken (2016)
8. Babar, M.A., Brown, A.W., Mistrik, I.: Agile Software Architecture Aligning Agile Processes and Software Architectures, 292p. Morgan Kaufmann, Burlington (2014)
9. Ford, N., Parsons, R., Kua, P.: Building Evolutionary Architectures, 272p. O'Reilly Media, Sebastopol (2017)
10. Gasevic, D., Djuric, D., Devedzi, V. (eds.): Model Driven Architecture and Ontology Development. Springer, Heidelberg (2006). https://doi.org/10.1007/3-540-32182-9
11. Bass, L., Weber, I., Zhu, L.: DevOps: A Software Architect's Perspective, 463p. Pearson Education, Inc., London (2015)
12. Bloomberg, J.: The Agile Architecture Revolution: How Cloud Computing, REST-based SOA, and Mobile Computing are Changing Enterprise IT, 278p. Wiley, Hoboken (2013)
13. van der Valk, H., Hunker, J., Rabe, M., Otto, B.: Digital twins in simulative applications: a taxonomy. In: Proceedings of the 2020 Winter Simulation Conference. https://www.researchgate.net/publication/341235159_A_Taxonomy_of_Digital_Twins. Accessed 5 May 2022
14. Digital Thread. https://searcherp.techtarget.com/definition/digital-thread. Accessed 5 May 2022
15. What is the Digital Thread? Digital Thread Definition. https://nxrev.com/2018/05/digitalthread/. Accessed 5 May 2022
16. Rozanski, N., Woods, E. (eds.): Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives, 576p. Viewpoints, Poland (2005)
17. Erder, M., Pureur, P., Woods, E.: Continuous Architecture in Practice Software Architecture in the Age of Agility and DevOps, Upper Saddle River, NJ, USA, 321p. (2021)
18. Shaw, M., Garlan, D. (eds.): Software Architecture: Perspectives on an Emerging Discipline, 242p. Prentice-Hall Inc., Hoboken (1996)

19. Vodyaho, A.I., Zhukova, N.A., Shichkina, Y.A., Anaam, F., Abbas, S.: About one approach to using dynamic models to build digital twins. Designs **6**, 25 (2022)
20. Osipov, V., Stankova, E., Vodyaho, A., Lushnov, M., Shichkina, Y., Zhukova, N.: Automatic synthesis of multilevel automata models of biological objects. In: Misra, S., et al. (eds.) ICCSA 2019. LNCS, vol. 11620, pp. 441–456. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-24296-1_35
21. Osipov, V., Zhukova, N., Vodyaho, A.: About one approach to multilevel behavioral program synthesis for television devices. Int. J. Comput. Commun. **11**, 17–25 (2017)
22. Vodyaho, A., Osipov, V., Zhukova, N., Chernokulsky, V.: Data collection technology for ambient intelligence systems in Internet of Things. https://doi.org/10.3390/electronics9111846. Accessed 5 May 2022
23. Osipov, V., Lushnov, M., Stankova, E., Vodyaho, A., Zukova, N.: Inductive synthesis of the models of biological systems according to clinical trials. In: Gervasi, O., et al. (eds.) ICCSA 2017. LNCS, vol. 10404, pp. 103–115. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-62392-4_8
24. Capilla, R., Bosch, J., Kang, K.C. (eds.): Systems and Software Variability Management, 317p. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36583-6
25. Xing, L.: Cascading failures in Internet of Things: review and perspectives on reliability and resilience. IEEE Internet Things J. **8**, 44–64 (2020). https://doi.org/10.1109/JIOT.2020.3018687
26. Vodyaho, A., Zhukova, N., Subbotin, A., Anaam, F.: Towards dynamic model-based agile architecting of cyber-physical systems. Sensors **22**(8), 3078 (2022). https://doi.org/10.3390/s22083078
27. Subbotin, A., Zhukova, N., Man, T.: Architecture of the intelligent video surveillance systems for fog environments based on embedded computers. In: Proceedings of the 2021 10th Mediterranean Conference on Embedded Computing (MECO), Budva, Montenegro, 7–10 June 2022, pp. 1–8 (2022). https://doi.org/10.1109/MECO52532.2021.9460270