APPLIED COMPUTER SCIENCE

The Journal is a peer-reviewed, international, multidisciplinary journal covering a broad spectrum of topics of computer application in production engineering, technology, management and economy.

The main purpose of Applied Computer Science is to publish the results of cutting-edge research advancing the concepts, theories and implementation of novel solutions in computer technology. Papers presenting original research results related to applications of computer technology in production engineering, management, economy and technology are welcomed.

We welcome original papers written in English. The Journal also publishes technical briefs, discussions of previously published papers, book reviews, and editorials. Especially we welcome papers which deals with the problem of computer applications in such areas as:

- manufacturing,
- engineering,
- technology,
- designing,
- organization,
- management,
- economics,
- innovations,
- competitiveness,
- quality and costs.

The Journal is published quarterly and is indexed in: BazTech, Cabell's Directory, CNKI Scholar (China National Knowledge Infrastucture), ERIH PLUS, Google Scholar, Index Copernicus, J-Gate, Scopus, TEMA Technik und Management.

Letters to the Editor-in-Chief or Editorial Secretary are highly encouraged.

CONTENTS

Applied Computer Science, vol. 16, no. 4, pp. 5–20
doi:10.23743/acs-2020-25

Submitted: 2020-11-29
Revised: 2020-12-07
Accepted: 2020-12-17

software-defined networks, fog computing, smart sensors, IoT

Ihor PYSMENNYI [0000-0001-7648-2593]*,
Anatolii PETRENKO [0000-0001-6712-7792]*,
Roman KYSLYI [0000-0002-8290-9917]*

GRAPH-BASED FOG COMPUTING NETWORK MODEL

Abstract

IoT networks generate numerous amounts of data that is then transferred to the cloud for processing. Transferring data cleansing and parts of calculations towards these edge-level networks improves system's, latency, energy consumption, network bandwidth and computational resources utilization, fault tolerance and thus operational costs. On the other hand, these fog nodes are resource-constrained, have extremely distributed and heterogeneous nature, lack horizontal scalability, and, thus, the vanilla SOA approach is not applicable to them. Utilization of Software Defined Network (SDN) with task distribution capabilities advocated in this paper addresses these issues. Suggested framework may utilize various routing and data distribution algorithms allowing to build flexible system most relevant for particular use-case. Advocated architecture was evaluated in agent-based simulation environment and proved its' feasibility and performance gains compared to conventional event-stream approach

1. INTRODUCTION

Introduction of high-bandwidth mobile networks, portable sensors and actuators, energy-efficient microprocessors and communication protocols enabled wide adoption of Internet of Things (IoT) – an adaptive network connecting real-world things (including sensors which perceive the environment and actuators which may actively interact with it) between each other and the internet. Being lightweight

[*] National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Institute of Applied Systems Analysis, Department of System Design, 37, Peremohy ave., Kyiv, Ukraine, ihor.pismennyy@gmail.com, tolja.petrenko@gmail.com, kvrware@gmail.com

and portable the majority of these devices are resource-constrained in terms of computing and networking power by definition meaning the need for intermediate computing layer between them and the cloud (Rahmani, Liljeberg, Preden & Jantsch, 2018).

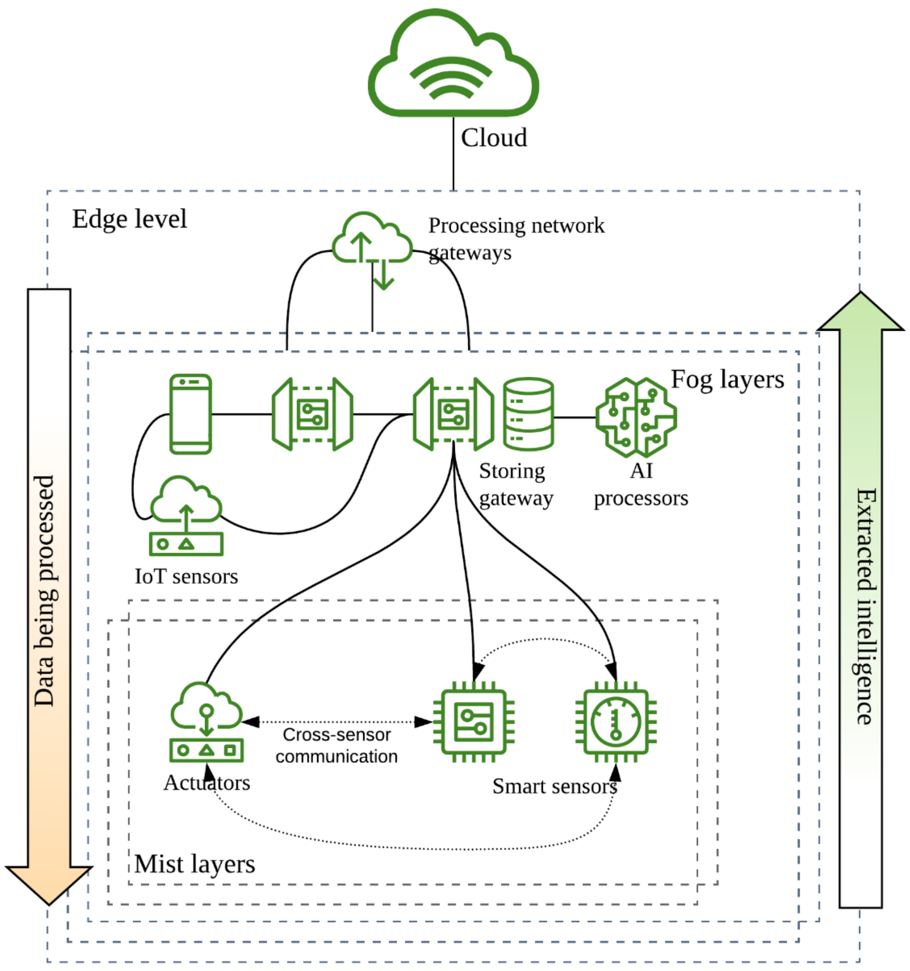


Fig. 1. Edge computing

As discussed in academia, this intermediate layer may utilize 3 complimentary approaches (Fig. 1):

1. Mist Computing. Computations are being performed on the extreme edge of the network – smart sensors and actuators themselves. Only pre-processed data is sent via network and IoT devices are not dependent on Internet connection (Yogi, Sekhar & Kumar, 2017).

2. Fog Computing. Layer close to the perception layer with computing, networking (aka gateway) and storage capabilities (Rahmani et al., 2018). Spans from the data creation point to its storage location allowing decentralized computing of gathered data. Any device with support of required network technologies, storage and networking capabilities can be utilized as fog a fog node (Joshi, n.d.). Can be considered as a superset among the mist layer.
3. Edge computing. (Satyanarayanan, 2017) defines any computing and network resource between data source and destination data centre (either cloud or local) as edge computing node. Further in this article edge-computing will be used as an umbrella term for all 3 levels.

Transferring part of data processing from cloud to edge level puts a lot of resource-related constraints such as computing power restrictions, absence of dynamic horizontal scalability and energy consumption limitations, but brings following benefits:

1. Location-awareness. Edge computing systems are aware of context in which computations are being performed
2. Latency reduction. Classical cloud-computing approach with aggregating data on *smart hub*, batch or stream sending it to cloud for processing and retrieving results back in synchronous or async manner introduces 2 pitfalls critical to real-time applications: network latency and possible network failure. Moving data processing to the edge can help to tackle these problems as discussed below and already found usage in various systems including
3. Security. Any data transmission is subject to man in the middle attacks and data protection requires energy-consuming encryption algorithms (Petrenko, Kyslyi & Pysmennyi, 2018a).
4. Eliminating bandwidth restrictions. Some data, especially media, require a high-bandwidth communication channel. By processing it on the edge level we eliminate the need for this expensive transfer. For example, smart doorbells which unlock the door using face identification tech may process video stream locally instead of sending it to cloud.
5. Energy consumption reduction. Data transfer is significantly more expensive in terms of energy consumption than basic processing. (Shi, Cao, Zhang, Li & Xu, 2016) hence offloading cleansing, aggregation and extraction operations to mist and fog layers may increase the time of autonomous work of IoT device as discussed in chapter 4.
6. Cost reduction. Edge computing helps utilize maximum of available resources resulting in improved cost-efficiency.

Modern edge computing use-cases and architectures are discussed in the next chapter with novel graph-based edge network architecture and its evaluation following.

2. LITERATURE REVIEW AND PROBLEM STATEMENT

Shifting distributed calculations paradigm from remote cloud to network edge is a complex task and involves solving novel architectural problems which can be grouped to resource constraints, communicational, privacy and security and fault tolerance domains. Further in this chapter state-of-the-art findings in the field are discussed.

(Ray, 2018) surveys and structures use-cases, technologies, and domains of modern IoT applications. Authors emphasize on technical challenges of designing service-oriented architecture of extremely heterogenous horizontally wide system, its' infrastructure and security requirements.

(Rahmani et al., 2018) provides a comprehensive high-level overview on most edge computing aspects with and various use-cases. Authors analyse architectural constraints, essential components, and management of fog-layer computational infrastructure. Edge computing has used in most parts of modern infrastructure from smart cities to medicine. In automotive cars already have sophisticated sets of various sensor and data processing systems allowing semi-autonomous driving (Hussain & Zeadally, 2019). Xiao & Zhu (2017) suggests using smart vehicles as moving fog nodes allowing on-demand computational resource distribution and expanding vehicle to vehicle (V2V) communications. These fog nodes may be connected to city infrastructure such as smart traffic lights allowing more efficient traffic distribution (Stojmenovic & Wen, 2014).

Chan, Estève, Escriba & Campo (2008) wraps a review of smart-home systems including permanent patient monitoring capabilities usually built on smart sensor networks. Gope & Hwang (2016) suggests Body Sensor Networks (BSN) architecture for distributed edge-level computations with regard to user's privacy. Data acquired from these wearable networks can be processed by deep convolutional neural networks on fog nodes for immediate anomaly detection (Petrenko, Kyslyi, & Pysmennyi, 2018b).

In use-cases mentioned above, critical security concerns are raised as health data is classified as sensitive (World Health Organization, 2010). In (Al Ameen, Liu, & Kwak, 2012) authors analyse privacy and security issues and requirements in regard to wireless sensor (WSN) and body area (WBAN) networks suggesting measures to cope with them. (Diogenes, 2017) suggests utilization of generic zoned approach with security boundaries for privacy preservation. Each of the *device zone, field gateway zone, cloud gateway zone, cloud services zone and remote users' zone* operates on constrained scope of user's data and has security requirements most suitable to given context, sensitivity of data being processed and persistence requirements. (Petrenko et al., 2018b) takes the idea further to cloud level allowing secure multi-party computations between akin organizations with help of hyperledger.

With big amount of open-source and proprietary communication solutions development of edge network on heterogeneous hardware is restrained by the absence of wide adopted standard open messaging protocol specification for exchanging and processing structured data. In (Kharchenko & Beznosyk, 2018) authors concentrate on building a unified data-flow format for various IoT devices suggesting JSON-based format for data and processing description. This approach enables proper distribution of computational resources and is essential for edge level information processing systems.

With absence of permanent remote monitoring capabilities concept of *self-awareness* become particularly important. According to (Rahmani et al., 2018) self-aware system has following capabilities:

1. Understands its current context and evaluates it to the desired state of the environment.
2. Understands its own state, monitors it to detect possible malfunctions and deviations.
3. Input data-aware – tracks its changes over history, performs semantic attribution and interpretation mapping properties to desirability scale.

(Lewis, Platzner, Rinner, Tørresen & Yao, 2016) introduces notion of *collective self-aware systems* where there is no central ‘awareness’ node emphasizing increased robustness and adaptability brought up by decentralized approach. Classification of awareness levels is also defined in book mentioned above. According to it, edge layer belongs to interaction-aware systems group as it acquires meaningful data from stimuli acting upon it and interactions with other systems, environment, and itself.

IEEE has recently adopted reference architecture for fog computing implementation for both hardware and software (IEEE Communications Society, 2018). Each fog node is operated by software backplane layer orchestrating internal (thing-to-fog) and external (fog-to-fog, fog-to-cloud) communications via service discovery, state management and pub-sub mechanisms, additionally enforcing authentication and system integrity.

Due to extremely distributed and heterogenous nature fog networks and absence of dynamic horizontal scalability vanilla service-oriented software architecture approach doesn’t work, thus some ideas may be refurbished. (Oma, Nakamura, & Duolikun, 2019) advocates a fault-tolerant tree-based fog network model. Each node calculates input from data obtained from one or more child nodes with sensors being leaves. System has process-transfer capabilities for fault-recovery and tree balancing to support dynamic network topology. This approach disregards variety of computational power between different nodes and certain elements of the system may become overloaded and create bottlenecks due to the hierarchical topology. Another disadvantage is the need for each node to know its downstream network and ancestors for the recovery mechanism effectively meaning need to persist the whole network structure in each of its element.

In addition to security and storage overhead critical for lightweight edge computing this approach requires frequent propagation of changes to the network topology unacceptable for dynamic fog networks. In further research authors suggested dynamic network-based fog computing (DNFC) model with auction method used to determine set of source nodes by each target node. Authors suggest broadcasting request to compute data from source node to possible fog computing targets, if target has enough energy to process given block or its part, target sends back acknowledgement, performs calculation and sends result upstream the network. This approach has clear advantage in case if all nodes are directly connected but does not support heterogeneous multi-layered smart sensor networks and does not regard potential overflow of communicational capacity of fog nodes.

In this paper we focus on development and evaluation of novel Graph-Based Fog Computing Network Model (GBFCNM) aimed to cope with issues mentioned above.

3. THE AIM AND OBJECTIVES OF THE STUDY

The purpose of this research is to analyse edge computational systems' architecture and propose a lightweight and flexible approach for distributing data processing among fog nodes. Given approach should support following capabilities:

1. Fault-tolerance – fog nodes may accidentally become inactive due to various reasons, for instance, low power supply, loss of communication channel and environmental situation.
2. Malfunction detection – fog nodes should be able to detect faulty behaviour of other node and have a recovery strategy for such cases.
3. Energy-efficiency – smart sensors are frequently designed wearable form-factor and thus have limited power supply.

Anylogic simulation environment is used to evaluate suggested architecture.

4. SMART-SENSOR DATA STREAM PROCESSING ARCHITECTURE

Traditional approach for processing data stream, including IoT generated values streams, is lambda-architecture (Fig. 2), which unifies real-time and historical batch analyses within the single framework (Marz & Warren, 2015). Data streams are ingested to message queue (or other data source) and then:

1. Processed via speed processing layer. Results are provided in real-time in synchronous (via responses to published API calls) or asynchronous manner (via exposing dedicated read API).

2. Batch-processing layer: raw data is being stored in Data Lake and then processed and stored to some sort of data warehouse to be analysed by batch jobs on schedule or on-demand.

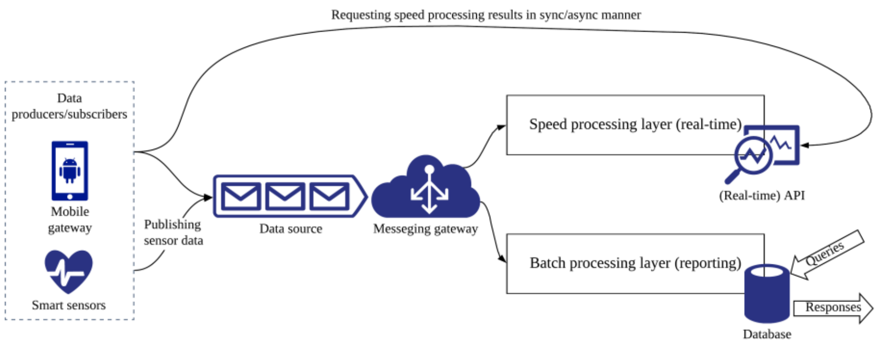

Requesting speed processing results in sync/async manner
Data producers/subscribers
Mobile gateway
Smart sensors
Publishing sensor data
Data source
Messeging gateway
Speed processing layer (real-time)
(Real-time) API
Batch processing layer (reporting)
Queries
Responses
Database

Fig. 2. Processing smart sensor data in conventional lambda architecture

Introducing edge computing brings following modifications to given data flow (Fig. 3):

1. Speed processing layer is moved from cloud to the network edge. Processing results may also be transferred to cloud for persistence. In addition to latency, load and security benefits this approach significantly reduces operational costs eliminating need for always-on cloud operation which allows using spot instances with dynamic pricing(“Spot Instances – Amazon Elastic Compute Cloud,” n.d.) and less strict fault-tolerance requirements which may result in abolishing infrastructure redundancy.
2. Data is transferred to cloud in pre-processed state eliminating the need for data lake and significantly reducing batch processing layer’s load by definition.

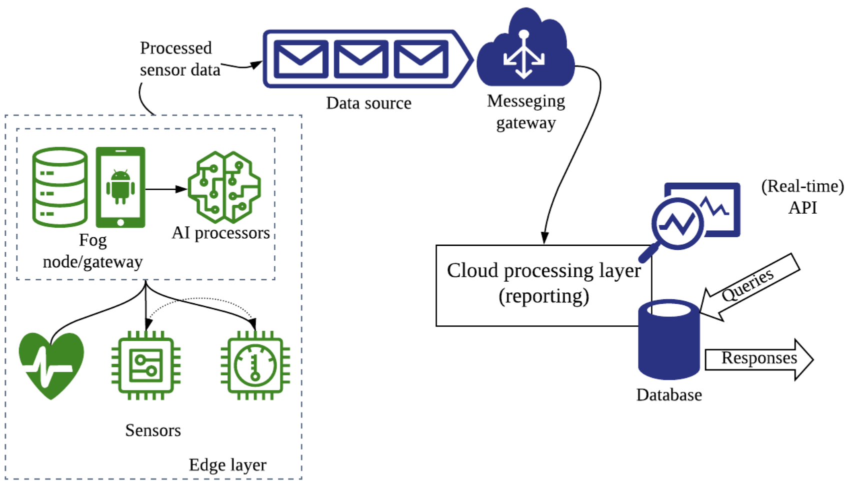


Fig. 3. Edge-optimized lambda architecture

Fog-layer is very heterogeneous as nodes may vary in terms of computational power and connectivity, thus need for dynamic and fault-tolerant computing model emerge. Proposed model is based on Software Defined Networking (SDN) principle: data flows within the network are performed and configured via the use of standardized application programming interfaces (API) (Kirkpatrick, 2013). By definition, GBFCNM is aimed to effectively process generated data. For the sake of simplicity let us assume processing time as the only effectiveness criteria and divide components of edge network to 3 groups based on their capabilities:

1. Smart sensor/actuator:
   - Emitting data and performing interactions with environment.
   - Internal (inside fog network), may be not directly connected to internet.
   - Low computational power as a trade-off for size and energy efficiency.
2. Fog node:
   - Internal traffic.
   - High computational power.
3. Gateway:
   - Accepts and redirects internal traffic.
   - Transmitting data to the cloud.
   - May also have high computational power.
4. Cloud – external component, destination of data.

Most edge-node devices utilize wireless radio network protocols including Bluetooth Low Energy, NFC, 6LoWPAN, Wi-Fi, ZigBee, and RFID, which have limited connection range. Therefore, each node N_x is assumed as connected to node N (belonging to its group G_N) if the distance between them is smaller than defined Δ:

$$N_x \in \{G_N\}: d(N, N_x) < \Delta \tag{1}$$

Proximity-based network partitioning is already adopted in various distributed event-based systems. Approaches with assigned brokers and dynamic plane are utilized (Castro-Jul et al., 2017). Fog network architecture suggested in this paper is classified as Distributed Control WSN as no central routing decision points are present and all nodes exchange information using dynamic data flow defined by this information itself and available processing capabilities (Mostafaei & Menth, 2018).

Each node advertises a set of available resource for each capability. As nodes are heterogeneous in terms of their capabilities it makes sense to have the possibility to transfer tasks between them (given assumption will be evaluated in following chapter), therefore fog node should also expose capabilities of its connections. As all network communications require time, computational resources, and energy, it is proposed to use penalty coefficient μ for each data transfer. In addition, this would limit number of task split (mapping) operations. So each fog node advertises following list of resources where N_i is identifier of resource owning node ($i = 0$ for current node, $i > 0$ for each connected node), Cap_j is capability category (e. g. "outbound traffic, kb/sec") and K_i is capacity of specified category available (e. g. 256 kb/sec):

$$\left[\{N_i, Cap_j, \alpha * K_i\}, \dots\right], \alpha = \begin{cases} 1, i = 0 \\ \mu, i \neq 0 \end{cases} \tag{2}$$

With this scenario it is easy to see that that after some time, when each service will send all advertisements (O(n^2) complexity where n is number of network nodes) with each request all current network topology will be sent, which is unfeasible in big multi-layered fog networks. To avoid this situation, it is proposed that each node advertises resources with maximum relative depth D. Further in this paper coefficient D=1 for the sake of simplicity (Fig. 4).

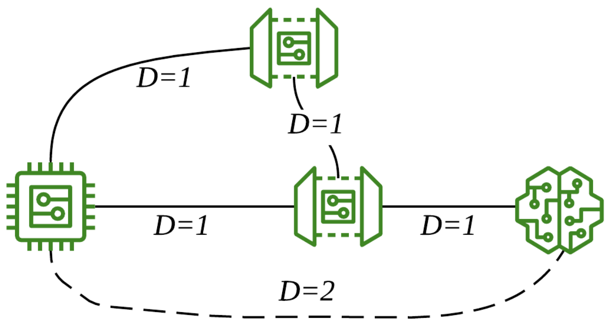


Fig. 4. Example fog network topology

Advertiser sends its' capabilities data at fixed schedule. Task owner node (source) receives set of capabilities from connected advertisers and selects target from recent targets (resources no older than given threshold τ) using multi-objective optimization methods (Fig. 6). Pareto efficiency with value weights calculated dynamically for each request based on task distribution (calculation or transfer) within it is used in this paper. If Pareto frontier consists of more than one element – random option is chosen. Timeout threshold minimizes connection attempts to no longer available nodes.

Suppose node N_0 requires computations in set of categories Cap (e. g. processing, cloud transfer) and has set of available resources from itself R_0. R_0^{Cap} is set of available resources in required categories. To process the required data N_0 received "offers" from nodes $N_i \dots N_j$ with corresponding available resources $R_i \dots R_j$ within τ time window. As N_0 is also a fog node it also advertised its capabilities, so $R_0 \in R_{i..j}$:

$$W(N) = \left(\sum_{Cap} \sum_{1..n} K\right) - \sum_{Cap} K0, \tag{3}$$

where N is number of connections (may be unique for each node).

Once data is processed it is being delivered to closest node with cloud-sending capabilities. Feasible path of distinct nodes towards the output node is called admissible path and in conventional SDNs is performed via SDN controller (Agarwal, Kodialam & Lakshman, 2013). As for networks are dynamic and decentralized, this node is selected via eager path searching algorithm (Sedgewick & Wayne, 2011). This approach requires trail of visited nodes' unique identifiers to be sent along with each request.

Due to the dynamic nature of edge-level networks their fragmentation is very possible, different network parts may become isolated and unable to transmit data further (Fig. 5).

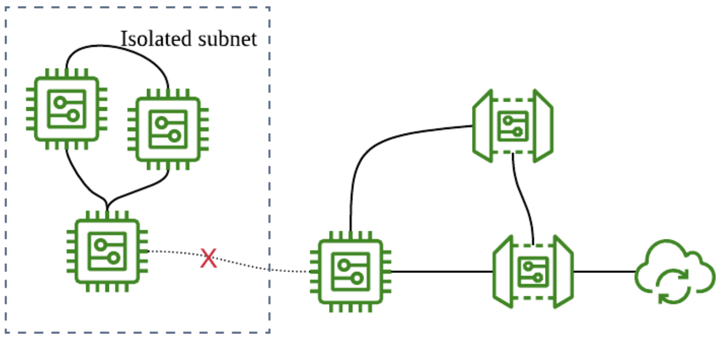


Fig. 5. Network fragmentation in fog computing

To overcome such situation asynchronous acknowledgement downstream is suggested (Fig. 6). Each node with storage capabilities should cache processed data stream. Node may erase cache only once it received acknowledgement that this data was processed upstream and after forwards it downstream. If this acknowledgement was not received within given timeout retry mechanism should be implemented.

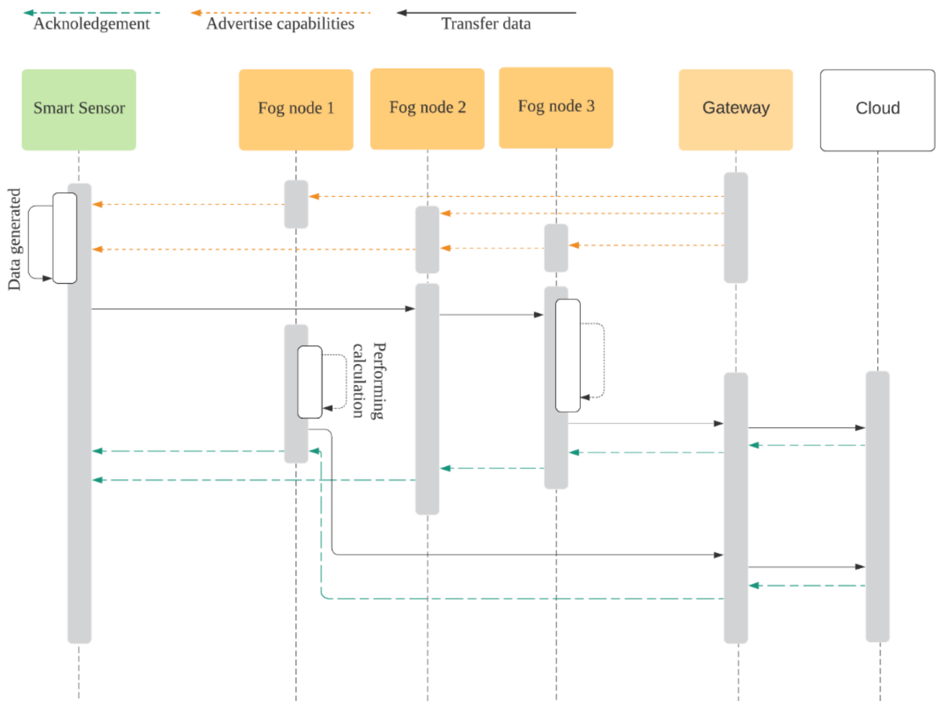


Fig. 6. GBFCNM happy flow

5. GBFCNM PERFORMANCE EVALUATION

IoT consist of the wide variety of interconnected devices with different power and capabilities. Testing and evaluating large-scale configurations of heterogeneous adaptive networks is almost unfeasible with conventional approaches, therefore utilization of multi-agent-based computing is suggested (Laghari & Niazi, 2016).

Each element of the system as well as the environment is modelled as an agent with dedicated behavioural and communicational strategy and capabilities creating system's digital representation in simulated environment (Klügl & Bazzan, 2012). Such approach allows us to evaluate behavioural strategies in various scenarios, identify their potential pitfalls, and determine optimal policies which will be implemented in end-product for different network configurations. There a lot of simulation toolkits available among which AnyLogic demonstrates significant usage growth dynamics and maintains vivid community (Dias, Vieira, Pereira & Oliveira, 2016). It allows combining agent based, discrete event and system dynamics simulations to single multi-method model. GBFCNM architecture was evaluated in Anylogic agent-based simulation environment (Fig. 7). Each actor (smart sensor, fog node, gateways and remote cloud) has dedicated set of state charts, behaviours and parameters which may dynamically change over time ("Multimethod Simulation Modeling for Business Applications – AnyLogic Simulation Software," n.d.). The goal of simulated system is to process and transfer data streams from smart sensors to cloud via fog and gateway nodes. Each node type has unique combination of capabilities as described above in system architecture chapter.

In (Table 1) conventional IoT setup (computations being performed in cloud only) with 4 smart sensors, 2 fog computing nodes acting as internal traffic routers and single gateway is evaluated against same setup with edge-computing capabilities. Each smart sensor emits N packets per second, which may be processed and forwarded by fog nodes reducing amount to be sent by F%. Processing rate of such node is K packets per second. Gateway is able to send to cloud L packets per second.

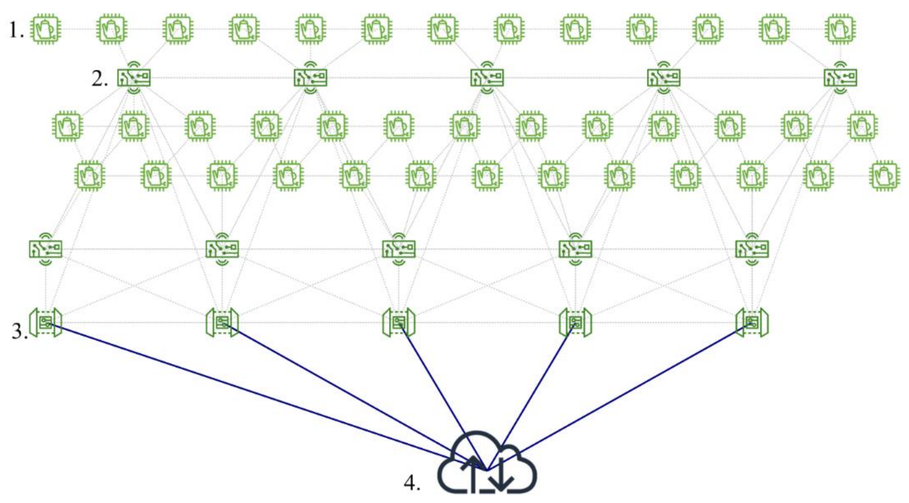


Fig. 7. Screenshot of simulated fog network consisting of smart sensors (1), fog nodes (2), gateways (3) and remote cloud (4)

6. DISCUSSION

The downside of given approach is the need to advertise capability information on connected nodes and append, which consumes network resources of the system. Possible solution is to add threshold after which resource is considered too weak and is no longer advertised. Developing strategies of route caching will allow decreasing configuration sharing frequency.

In addition, utilization of generic frontier search algorithm for routing may significantly increase message size in large distributed systems as path is sent together with message. Introduction of dedicated control nodes forming indirectly (via cross-sensor multi-hop communications) connected SDN controller may help solving this issue and is a subject for further research (Mostafaei & Menth, 2018).

Possible loss of data if some subset of nodes become unavailable due to dynamic nature of the fog network is tackled with its' caching on intermediate nodes. Fog network nodes have limited storage capabilities (for example, most STM32 boards come with 32 KB to 1MB flash memory), therefore utilization of data compression techniques is suggested. In (Pysmennyi, Kyslyi, & Petrenko, 2019) authors advocate using moving average on the oldest stored data windows, so most up-to-date information will still have highest possible resolution:

$$CMA_n = \frac{x_1 + \cdots + x_n}{n};\ CMA_{n+1} = \frac{x_{n+1} + n * CMA_n}{n+1}. \tag{4}$$

Suggested approach also allows to tackle spikes of volume of generated data if it exceeds network bandwidth.

Another issue of suggested algorithm is gradient load of computational resources – the further they are from data emitting smart sensor the less they would be loaded due to data transfer penalties. With generic fog network where sensors are distributed evenly with fog nodes this will not pose a problem, but in case of network with homogenous resource clusters this approach requires adjustments which are subject for the further research.

7. CONCLUSIONS AND FUTURE WORK

As shown in this paper, SDN paradigm perfectly fits fog computing networking tasks. Utilization of edge nodes for computations clearly showed advantages in decreasing load of cloud system, improved resource utilization on the edge level and enabled fast feedback to actuator nodes. Power, computational and bandwidth constraints combined with narrow scope of capabilities (specialization of nodes) introduce the need for flexible framework for distributed processing and routing of data suggested in this paper.

As shown in simulation above, given approach results in significant reduction of bandwidth and computational load to cloud infrastructure and improves overall system efficiency.

Possible areas of future research include but are not limited to:

1. Adaptive selection of most suitable mesh networking algorithm. Given challenge implicitly requires enabling monitoring of low-powered distributed system.
2. Introduction of SDN control plane to the system for improvement of network efficiency.
3. Combining routing with processing task mapping for increasing load distribution efficiency.
4. Faulty and malicious nodes detection. Evaluation of using AI-empowered fog nodes for this purpose.

REFERENCES

Agarwal, S., Kodialam, M., & Lakshman, T. V. (2013). Traffic engineering in software defined networks. *2013 Proceedings IEEE INFOCOM*, 2211–2219. https://doi.org/10.1109/INFCOM.2013.6567024

Al Ameen, M., Liu, J., & Kwak, K. (2012). Security and privacy issues in wireless sensor networks for healthcare applications. *Journal of Medical Systems, 36*(1), 93–101. https://doi.org/10.1007/s10916-010-9449-4

Castro-Jul, F., Conan, D., Chabridon, S., Díaz Redondo, R. P., Fernández Vilas, A., & Taconet, C. (2017). Combining Fog Architectures and Distributed Event-Based Systems for Mobile Sensor Location Certification. *Lecture Notes in Computer Science, 10586,* 27–33. https://doi.org/10.1007/978-3-319-67585-5_3

Chan, M., Estève, D., Escriba, C., & Campo, E. (2008). A review of smart homes-Present state and future challenges. *Computer Methods and Programs in Biomedicine*, *91*(1), 55–81. https://doi.org/10.1016/j.cmpb.2008.02.001

Dias, L. M. S., Vieira, A. A. C., Pereira, G. A. B., & Oliveira, J. A. (2016). Discrete simulation software ranking — A top list of the worldwide most popular and used tools. *2016 Winter Simulation Conference (WSC)*, 1060–1071. https://doi.org/10.1109/WSC.2016.7822165

Diogenes, Y. (2017). Internet Of Things Security Architecture. Retrieved December 31, 2018, from Microsoft website: https://docs.microsoft.com/en-us/azure/iot-fundamentals/iot-security-architecture

Gope, P., & Hwang, T. (2016). BSN-Care: A Secure IoT-Based Modern Healthcare System Using Body Sensor Network. *IEEE Sensors Journal*, *16*(5), 1368–1376. https://doi.org/10.1109/JSEN.2015.2502401

Hussain, R., & Zeadally, S. (2019). Autonomous Cars: Research Results, Issues, and Future Challenges. *IEEE Communications Surveys and Tutorials*, *21*(2), 1275–1313. https://doi.org/10.1109/COMST.2018.2869360

IEEE Communications Society. (2018). IEEE Standard for Adoption of OpenFog Reference Architecture for Fog Computing. In *The Institute of Electrical and Electronics Engineers*. https://doi.org/10.1109/IEEESTD.2018.8423800

Joshi, N. (n.d.). Fog vs Edge vs Mist computing. Which one is the most suitable for your business? Retrieved June 21, 2020, from https://www.allerin.com/blog/fog-vs-edge-vs-mist-computing-which-one-is-the-most-suitable-for-your-business

Kharchenko, K., & Beznosyk, O. (2018). The input file format for IoT management systems based on a data flow virtual machine. *2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT)* (139–142). IEEE. https://doi.org/10.1109/DESSERT.2018.8409115

Kirkpatrick, K. (2013). Software-defined networking. *Communications of the ACM*, *56*(9), 16–19. https://doi.org/10.1145/2500468.2500473

Klügl, F., & Bazzan, A. L. C. (2012). Agent-Based Modeling and Simulation. *AI Magazine*, *33*(3), 29. https://doi.org/10.1609/aimag.v33i3.2425

Laghari, S., & Niazi, M. A. (2016). Modeling the Internet of Things, Self-Organizing and Other Complex Adaptive Communication Networks: A Cognitive Agent-Based Computing Approach. *PLOS ONE*, *11*(1), e0146760. https://doi.org/10.1371/journal.pone.0146760

Lewis, P. R., Platzner, M., Rinner, B., Tørresen, J., & Yao, X. (2016). Self-aware Computing Systems. In P. R. Lewis, M. Platzner, B. Rinner, J. Tørresen, & X. Yao (Eds.), *Natural Computing Series*. https://doi.org/10.1007/978-3-319-39675-0

Marz, N., & Warren, J. (2015). *Big Data: Principles and best practices of scalable realtime data systems* (1st ed.). Manning Publication.

Mostafaei, H., & Menth, M. (2018). Software-defined wireless sensor networks: A survey. *Journal of Network and Computer Applications*, *119*(June), 42–56. https://doi.org/10.1016/j.jnca.2018.06.016

Multimethod Simulation Modeling for Business Applications – AnyLogic Simulation Software. (n.d.). Retrieved October 5, 2020, from https://www.anylogic.com/resources/white-papers/multimethod-simulation-modeling-for-business-applications/

Petrenko, A., Kyslyi, R., & Pysmennyi, I. (2018a). Designing security of personal data in distributed health care platform. *Technology Audit and …*, *2*(42). https://doi.org/10.15587/2312-8372.2018.141299

Petrenko, A., Kyslyi, R., & Pysmennyi, I. (2018b). Detection of human respiration patterns using deep convolution neural networks. *Eastern-European Journal of Enterprise Technologies*, *4*(9(94)), 6–13. https://doi.org/10.15587/1729-4061.2018.139997

Pysmennyi, I., Kyslyi, R., & Petrenko, A. (2019). Edge computing in multi-scope service-oriented mobile healthcare systems. *System Research and Information Technologies*, (1), 118–127. https://doi.org/10.20535/SRIT.2308-8893.2019.1.09

Rahmani, A. M., Liljeberg, P., Preden, J.-S., & Jantsch, A. (2018). *Fog Computing in the Internet of Things*. Springer. https://doi.org/10.1007/978-3-319-57639-8

Ray, P. P. (2018). A survey on Internet of Things architectures. *Journal of King Saud University - Computer and Information Sciences*, *30*(3), 291–319. https://doi.org/10.1016/j.jksuci.2016.10.003

Oma, R., Nakamura, S., & Duolikun, D. (2019). A fault-tolerant tree-based fog computing model. *International Journal of Web and Grid Services*, *15*(3), 219. https://doi.org/10.1504/IJWGS.2019.10022420

Satyanarayanan, M. (2017). Edge Computing. *Computer*, *50*(10), 36–38. https://doi.org/10.1109/MC.2017.3641639

Sedgewick, R., & Wayne, K. (2011). Algorithms. In *Foreign Affairs* (4th ed.). Westford: Addison-Wesley.

Shi, W., Cao, J., Zhang, Q., Li, Y., & Xu, L. (2016). Edge Computing: Vision and Challenges. *IEEE Internet of Things Journal*, *3*(5), 637–646. https://doi.org/10.1109/JIOT.2016.2579198

Spot Instances – Amazon Elastic Compute Cloud. (n.d.). Retrieved July 7, 2020, from https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-spot-instances.html

Stojmenovic, I., & Wen, S. (2014). *The Fog Computing Paradigm: Scenarios and Security Issues*. *2*, 1–8. https://doi.org/10.15439/2014F503

World Health Organization. (2010). Telemedicine Opportunities and developments in Member States. In *World Health Organization* (Vol. 2).

Xiao, Y., & Zhu, Ch. (2017). Vehicular fog computing: Vision and challenges. *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, 6–9. https://doi.org/10.1109/PERCOMW.2017.7917508

Yogi, M. K., Sekhar, K. C., & Kumar, G. V. (2017). Mist Computing: Principles, Trends and Future Direction. *International Journal of Computer Science and Engineering*, *4*(7), 19–21. https://doi.org/10.14445/23488387/IJCSE-V4I7P104

Applied Computer Science, vol. 16, no. 4, pp. 21–36
doi:10.23743/acs-2020-26

Submitted: 2020-11-04
Revised: 2020-11-24
Accepted: 2020-12-02

Forecasting, ARIMA, Inventory management, Lot-sizing, Economies-of-scale, Production planning, Heuristic

Jack OLESEN[*], Carl-Emil Houmøller PEDERSEN[*], Markus Germann KNUDSEN[*], Sandra TOFT[*], Vladimir NEDBAILO[*], Johan PRISAK[**], Izabela Ewa NIELSEN[*], Subrata SAHA[*]

JOINT EFFECT OF FORECASTING AND LOT-SIZING METHOD ON COST MINIMIZATION OBJECTIVE OF A MANUFACTURER: A CASE STUDY

Abstract

Forecasting and lot-sizing problems are key for a variety of products manufactured in a plant of finite capacity. The plant manager needs to put special emphasis on the way of selecting the right forecasting methods with a higher level of accuracy and to conduct procurement planning based on specific lot-sizing methods and associated rolling horizon. The study is conducted using real case data form the Fibertex Personal Care, and has evaluated the joint influence of forecasting procedures such as ARIMA, exponential smoothing methods; and deterministic lot-sizing methods such as the Wagner-Whitin method, modified Silver-Meal heuristic to draw insights on the effect of the appropriate method selection on minimization of operational cost. The objective is to explore their joint effect on the cost minimization goal. It is found that a proficient selection process has a considerable impact on performance. The proposed method can help a manager to save substantial operational costs.

[*]Aalborg University, Department of Materials and Production, , DK 9220, Aalborg East, Denmark, subrata.scm@gmai.com, saha@m-tech.aau.dk
[**] Production Manager, Fibertex Personal Care Group, Aalborg, Denmark

1. INTRODUCTION

In a pragmatic scenario, there exists a natural link between forecasting accuracy and inventory replacement decision. If the demand for a product is at the higher side compared to the expected, i.e. estimated through scientific forecasting tool or educated guessing, a firm can face a stock-out, and when the estimated demand is below the actual level, the firm needs to incur additional holding or operational costs. On the other hand, if a firm makes orders frequently, the policy can reduce holding costs at the expense of higher ordering costs but can face stockout situations. This problem necessitates the integration of the use of robust statistical forecasting methods and lot-sizing decisions. However, many organizations still count on the judgmental adjustment based approach by stock-keeping unit managers for both slow- and fast-moving products (Fildes, Goodwin, Lawrence & Nikolopoulos, 2009). Moreover, researchers and practitioners considered the issue of accurate demand forecasting and inventory lot-sizing decision as two independent decision-making processes, and without the integration of these two-decision processes, it can lead to a suboptimal outcome for a firm (Syntetos, Nikolopoulos & Boylan, 2010). Recently, a project for a Danish-based company Fibertex Personal Care (FPC) was undertaken, which is owned by the Danish conglomerate Schouw & Co. to explore the joint performance of scientific forecasting methods and lot-sizing formulas for time-varying demand (Pedersen et al., 2020).

A single-item, single-level, incapacitated economic lot-size problem with constant cost parameters, time-varying demand rate, and discrete opportunities for replenishment is assumed. Note that the dynamic lot-sizing models under a deterministic environment address the problem of finding an optimal production or replacement planning to minimize total cost that includes fixed setup and holding cost over the time horizon (Silver & Meal, 1973; Van Den Heuvel & Wagelmans, 2005; Saha, Das & Basu, 2010; Grubbström & Tang, 2012; Eriksen & Nielsen, 2016; Moon, Yoo & Saha, 2016; Nilakantan, Li, Tang & Nielsen, 2017, Kian et al., 2020; Ho & Ireland, 2012). A fixed order cost is incurred for each order and holding costs incur for each unused unit stored in each period (Drexl & Kimms, 1997). Although there exist several lot sizing techniques, the Wagner-Whitin (WW) method has been extensively preferred because it can provide optimal outcomes (Heady & Zhu, 1994). When determining the most suitable forecasting performance measures, the product characteristics and inventory management should be taken into consideration, since the objectives of forecasting and inventory control usually are inconsistent. Xi et al. (2012) also study the effect of linkage between forecasting and inventory management, and found that the traditional forecasting performance measures decrease in performance without proper link.

Inventories are considered to be one of the key assets of an organization, the size of inventory can be determined through different forecasting techniques (Silver, Pyke & Thomas, 2016). Inaccurate forecasts turn out to be expensive for organization operations, in terms of overstocking or stock-outs and lost sales, while the desired service level is not being met (Kourentzes, Trapero & Barrow, 2020). Inventory planning mainly focuses on when to order and how much to order, the lot sizes. Lot-size in the context of this study represents the purchased in a single transaction, while inventory lot-sizing involves determining and scheduling lot sizes, so demand is satisfied in each period of the planning horizon. Optimization of inventory lot-sizing refers to minimizing the total inventory cost, by having a trade-off between large production lots, resulting in low ordering costs, and lot-for-lot ordering resulting in low holding costs. Andriolo et al., (2014) classify lot-sizing into three different models; deterministic, stochastic, and fuzzy models. Several extensions of inventory lot-sizing exists, but it consists of a fixed or variable order quantity together with the periodic or continuous frequency of review. The underlying parameters include: finite or infinite horizon, single or multiple items, deteriorating or not, zero or non-zero fixed or varying lead time, capacitated or incapacitated, deterministic, time-varying or constant, or stochastic demand, single- or multi-echelon, back-ordering or not, fixed or rolling planning horizon, with or without quantity discounts and with constant or fuzzy cost parameters.

Many manufacturing firms feel pressured to cut costs and improve profitability because of increasing competition and globalization (Świć & Gola, 2013). In this regards, business system analytic are designed to facilitate the flow of information and decent planning. However, those systems if not managed carefully, can result in conflict and degrade performance. Researchers pointed out that, a group of organizations are still facing implementation issues; many others fear implementation because of the costs and the pros and cons of implementation (Patalas-Maliszewska, 2012). The most common causes of business system analytic failures are a combination of high software customization combination, poor planning and commitment, relying on legacy systems, lack of clarity about required changes etc. The objective is to explore answer to the following research question: Does there any scope to reduce cost by improving inventory planning and forecasting support systems? Therefore, daily usages data was collected for the 14th month for one product from the FPC, namely Spunbond PP. Several exponential smoothing methods and ARIMA was employed for forecasting requirements. For a lot-sizing decision, the WW method, and its coherent heuristics like Silver-Meal (SM) heuristic (Baker, 1989), modified Silver-Meal (MSM) heuristic, and EOQ heuristic was used. It was found that the company is struggling to achieve desirable outcome through their existing business analytic framework. Compare to existing literature where, forecasting and procurement planning are mainly considered as independent decisions; this study evaluates their joint impact on system wide performance. Therefore, the insights can help them to improve the decision making process. The difference between the problem investigated in this study from those in the existing lot-sizing and forecasting problem is that our focus on the issue of actual implementation in the presence of opportunities to exploit information and scale economies. Our findings suggest that the performance of lot-sizing algorithms appear with different magnitudes and the outcomes leads to a desirable for a short forecast horizon, and which suggest that planning for long period does not necessarily result in a good planning.

2. METHODS

In this section, an overview of forecasting methodology and lot-sizing techniques used in this study is described.

2.1. Forecasting methods

In this study, two classes of forecasting methods was used: (i) exponential smoothing, and (ii) ARIMA model. Forecasting accuracy always key in decision-making process (Bocewicz, Nielsen, Banaszak & Thibbotuwawa, 2018; Nielsen, Jiang, Rytter & Chen, 2014) and the comparative study will help the production managers to explore their impact.

2.1.1. Exponential-smoothing model used in this study

Exponential models used in the study are in of the form presented below:

$$y_t = a_t + b_t t + s(t) + \varepsilon_t, \tag{1}$$

where: a_t, b_t, and $s(t)$ represent the time-varying mean; time-varying slope; and time-varying seasonal component, respectively. In addition, A_t and B_t represent smoothed level that estimates a_t and b_t, respectively. S_{t-j}, $j = 0,\ldots, s-1$ estimates of the $s(t)$. The last components ε_t represent the exogenous random shocks. In addition, it is assumed that α, γ, and δ represent level, trend, and seasonal smoothing weight, respectively. Therefore, larger (small) weights ensure higher (lower) influence to newer observation. Based on the above notation, the methods used in this study is summarized in Table 1.

Tab. 1. Exponential forecasting methods used in this study

Method	Model equation
Simple	$y_t = a_t + \varepsilon_t$ $A_t = \alpha y_t + (1-\alpha)\, A_{t-1}$
Brown	$y_t = a_t + b_t t + \varepsilon_t$ $A_t = \alpha y_t + (1-\alpha)\, A_{t-1}$
Holt	$y_t = a_t + b_t t + \varepsilon_t$ $A_t = \alpha y_t + (1-\alpha)(A_{t-1} + B_{t-1})$ $B_t = \gamma(A_t - A_{t-1}) + (1-\gamma)B_{t-1}$
Seasonal	$y_t = a_t + s(t) + \varepsilon_t$ $A_t = \alpha(y_t - S_{t-s}) + (1-\alpha)A_{t-1}$ $S_t = \delta(y_t - A_{t-s}) + (1-\delta)S_{t-s}$
Winter additive	$y_t = a_t + b_t t + s(t) + \varepsilon_t$ $A_t = \alpha(y_t - S_{t-s}) + (1-\alpha)(A_{t-1} + B_{t-1})$ $B_t = \gamma(A_t - A_{t-1}) + (1-\gamma)B_{t-1}$ $S_t = \delta(y_t - A_{t-s}) + (1-\delta)S_{t-s}$

2.1.2. ARIMA

Time series data frequently experienced both trend and seasonal patterns and might be non-stationary in nature. Therefore, autoregressive integrated moving average (ARIMA(p,d,q)) models are widely used for forecasting (Mills, 2019), where *p*, *q*, and *d* are positive integer numbers, referring to the order of the autoregressive, moving average, and integrated parts of the model, respectively. In an ARIMA model, the future value of a variable is assumed a linear function of several past observations plus random errors. The linear function is based upon three parametric components: auto-regression (AR), integration (I), and moving average (MA). If *d* = 0, then the model reduces to an *ARMA* (*p*,*q*) model. If *p* = *q* = 0, then it simply converts to the Moving Average (*q*) model. In general, the ARIMA *(p,0,0)* model is formulated as follows:

$$y_t = c + \varphi_1 y_{t-1} + \cdots + \varphi_p y_{t-p} + \varepsilon_t, \tag{2}$$

where y_t and ε_t are the actual value and random error, assumed to be independently and identically distributed with a mean of zero and a constant variance of σ^2; at period t, respectively; *c* is the intercept(constant); φ_p are a finite set of parameters, determined by linear regression. Similarly, the MA (ARIMA *(0,0,q)*) model is formulated as follows:

$$y_t = c_1 - \theta_1\varepsilon_{t-1} - \cdots - \theta_q\varepsilon_{t-q} + \varepsilon_t, \tag{3}$$

φ_p is a finite set of parameters; and c_1 is the mean of the series. For the detailed discussion on ARIMA, one can see Box et al., (2011), Taneja et al., (2016).

2.1.3. Performance measure for forecasting

In this study, three performance measures is used to evaluate accuracy of forecast as follows:

- Mean absolute error (MAE) $= \frac{1}{n}\sum_{t=1}^{n} |y_t - f_t|$,
- Mean absolute percentage error (MAPE) $= \frac{1}{n}\sum_{t=1}^{n} |\frac{y_t - f_t}{y_t}|$,
- Root mean squared error (RMSE) $= \sqrt{\frac{1}{n}\sum_{t=1}^{n}(y_t - f_t)^2}$.

Note that f_t represents the forecasted value. For an accurate forecast, the value of MAE, MAPE, and RMSE should be as small as possible.

2.2. Lot-sizing method for time-varying demand

Lot-sizing decision under the time-varying demand is always a subject of importance to planning a robust replenishment decision (De Bodt, Gelders & Van Wassenhove, 1984). A comprehensive study was conducted of the relative performance among WW method (Wagner & Whitin, 1958), EOQ heuristic, SM and MSM heuristics (Silver & Miltenburg, 1984). The cost performance of each method was compared against the WW method by evaluating the percentage deviation from the minimum total cost to holding and ordering cost.

The following assumptions are made to study the impact of the discrete lot-sizing model for the purpose of simplicity:

1. Demand is discrete (weekly basis) and known from forecasting information in advance. Therefore, any considerations of nervousness or stochastic parameters are excluded. The requirements of each week must be available at the beginning of that period.
2. Requisitions are instantaneous; that is lead time is negligible. Shortages are not allowed.
3. The entire procured quantity is delivered at a time and benefits from joint replenishment are ignored.
4. Costs involved are inventory carrying cost and ordering cost. It is assumed that both units carrying cost per period and ordering cost are constant throughout the considered aggregated planning horizon and independent on the replacement quantity.

The following notation is used to describe lot-sizing methods:

Tab. 2. Notations

N	number of periods(weeks)
i	number of weeks, i ∈ {1··· ,N}
T	the number of periods for the planning
D_i	requirement at ith week (forecast)
H	Holding cost
A	Ordering cost

2.2.1. Economic order quaint (EOQ)

When the demand rate is approximately constant, a fixed EOQ model can be applied. However, when demand is a time-varying rate, one can ignore the variability by considering the average demand rate ($D = \frac{\sum_{i=1}^{T} D_i}{T}$) to calculate EOQ, and the EOQ is applied when a requisition is made. Furthermore, $\overline{D}$ can be based on an infrequent estimate of the average demand per period, and therefore, it is not necessary to reevaluate at each replenishment decision. Therefore, first, $EOQ = \sqrt{\frac{2AD}{H}}$ was computed and then at the time of a replenishment, the optimal EOQ is adjusted to exactly satisfy the requirements of a forthcoming integer number of consecutive periods to reduce the inventory holding cost (Silver et al., 2016).

2.2.2. Wagner-Whitin method

The classical WW method was developed to find an optimal ordering policy for deterministic and time-varying demand. The following formulation can be used to present the algorithm:

$$C(t) = min\left\{A + C(t-1), \min_{1 \le i \le t}\left\{A + C(t-1) + \sum_{h=i}^{t-1} \sum_{k=h+1}^{t} H_h D_k\right\}\right\} \quad (4)$$

where $C(0) = 0$, $C(1) = A$, and $C(t)$ represents the minimum cost of ordering and holding inventory for periods 1 to t. Note that a replenishment decision only takes place when the inventory level is zero. There is an upper limit to how far before a period t would be included its requirements (D_t) in a replenishment quantity. Eventually, the carrying costs become so high that it is less expensive to have a replenishment arrive at the start of period j than to include its requirements in a replenishment from earlier periods.

2.2.3. Silver-Meal Heuristic and its modification

The SM heuristic was designed to obtain an easy and effective way to obtain a replenishment strategy under deterministic time-varying demand (Silver and Meal, 1973). As mentioned by the authors, the heuristic is myopic in nature and the goal is to choose a replenishment quantity by minimizing costs per unit time only to the end of the period covered by the replenishment under consideration. Then, the basic idea is to select the lowest period of T by minimizing the following function:

$$C_T = \frac{A + H\sum_{t=1}^{T}(t-1)D_t}{TH}, T = 1,2,3, \ldots, \quad (5)$$

where C_T represents the normalized costs per unit time. Once a value of T is obtained, one needs to move the origin to the end of the period T and repeat the procedure to select the next replenishment interval. To improve the performance of the classical SM heuristic, several modifications are proposed by the researchers. The performance of one such modifications proposed by Silver and Miltenburg (1984) is used to eradicate the truncated horizon problem to some extent and reduce high-cost penalties.

3. CASE STUDY

The project is conducted with the Fibertex Personal Care (FPC) company and a subsidiary of Danish conglomerate Schouw & Co. FPC is one of the largest producers of Spunmelt Nonwovens for the hygiene industry. A regular visit was made to one of their production units at Aalborg, Denmark, to observe how the products are made and acquire detail knowledge about their production procedure (Pedersen et al., 2020). The company currently relies on their forecasting and lot-sizing method for a procurement decision and the regional staff sometimes need to take decision qualitatively, which can increase operational cost. Based on advice from specialists working on the FPC, first, one of their commonly used raw material procured from several suppliers is selected, so that it allowed us to provide a critical focus to explore ways to reduce holding costs and eradicate the possibilities of potential shortages by improving forecasting performance. The objective is to verify how the robust forecasting and lot-sizing methods can help them to eradicate the problem.

4. RESULT ANALYSIS AND DISCUSSION

First, an overview of daily usages of the raw material over the previous 58 weeks is presented. Note that the material is used every day (7 days) and the corresponding sequence plot is presented in Figure 1.

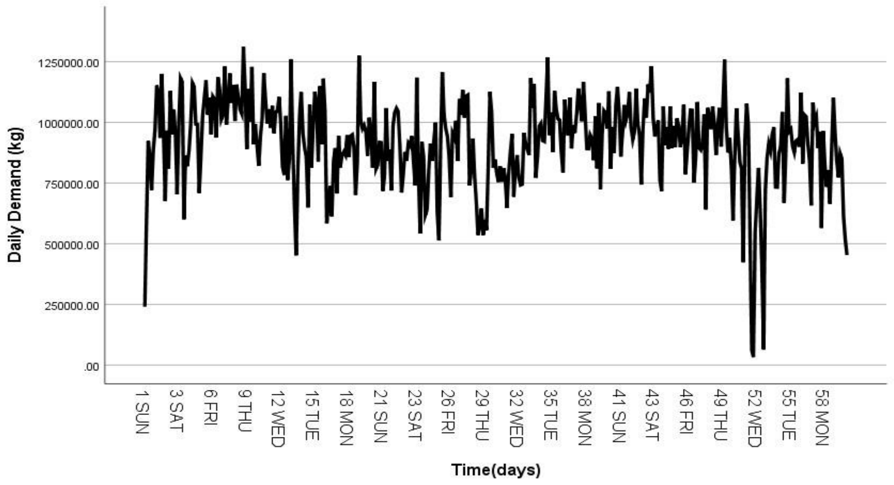


Fig. 1. Sequence plot of daily usage for the raw material

Figure 1 demonstrates that the daily usage almost follows a steady pattern, although the certain drop at week 52 is due to the Christmas Holiday. Based on the actual data, a forecast for the upcoming 14 weeks was made and results are presented below in Table 3.

Note that one cannot conclude about best forecasting methods, preference measures such as RMSE and MAE are higher for ARIMA (2,0,2), but MAPE remains high for ARIMA (0,2,2). Till we use results for ARIMA (2,0,2) for further analysis of determining performance lot-sizing methods. Statistical forecasting techniques have also advanced significantly; however, those have not been used extensively at an operational level mainly due to their complexity (Syntetos, Boylan & Disney, 2009). In our experience, simple exponential forecasting is sometime outperformed by ARIMA, but which might not be practiced while forecasting at the FPC. Next, an overview of lot-sizes based on the forecasted data was computed and presented in Table 4.

Tab. 3. Performance of various forecasting methods

Forecast for weekly requirement of SPUNBOND PP				
Method	Simple	Brown	Holt	Simple Seasonal
Stationary R^2	0.210	0.680	0.687	0.593
RMSE	164391.734	169147.082	164748.279	162363.999
MAPE	23.578	24.080	23.246	22.473
MAE	124825.799	128026.114	125627.671	123059.797

Parameter	α = 0.210	α = 0.106	α = 0.298
			γ = 0.00002
				α = 0.276
				δ = 0.095

Method	ARIMA (1,1,1)	ARIMA (1,1,0)	ARIMA (0,1,1)	ARIMA (1,0,1)
Stationary R^2	0.247	0.075	0.189	0.221
RMSE	159651.828	177013.880	165675.733	160860.753
MAPE	22.065	20.597	23.35	22.476
MAE	122226.078	136575.496	125573.318	121335.635

Parameter	C = -4701.925
	MA = 0.364
	AR = 0.922
		C = -5680.416
		AR = -0.274
			C = 1073.806
			MA = 0.743
				C = 907716.15
				AR = 0.536
				MA = 0.061

Method	ARIMA (2,2,2)	ARIMA (2,2,0)	ARIMA (2,0,2)	ARIMA (0,2,2)
Stationary R^2	0.693	0.417	0.247	0.57
RMSE	163266.4	224298.004	158505.502	192574.644
MAPE	21.823	24.454	22.363	20.233
MAE	125013.8	173679.233	119659.383	151371.789
Parameter	C = -35627.50 AR Lag 1 = 0.233 AR Lag 2 = -0.164 MA Lag 1 = 1.73 MA Lag 2 = -0.73	C = -23150.01 AR Lag 1 = -0.769 AR Lag 2 = -0.407	C = 901727.61 AR Lag 1 = 1.011 AR Lag 2 = -0.070 MA Lag 1 = 0.571 MA Lag 2 = 0.206	C = -19943.330 MA Lag 1 = 0.877 MA Lag 2 = 0.112
Method	ARIMA (2,0,0)	ARIMA (0,2,0)	ARIMA (0,0,2)	Winter Additive
Stationary R^2	0.22	0.01	0.211	0.593
RMSE	160874.79	292270.10	161919	162639.789
MAPE	22.46	29.13	22.83	22.447
MAE	121356.89	227842.62	122334.5	123234.070

Parameter	C = 907446.4 AR
	Lag 1 = 0.481
	AR Lag 2 = 0.019
		C = -43233.86	C = 905563.110
			MA Lag 1 = -0.478
			MA Lag 2 = -0.172
				α = 0.281
				γ = 0.001
				δ = 0.096

Tab. 4. Performance of different lot-sizing methods based on the forecasted data

Week	Requirement (kg)	EOQ	SM	WH	SMM
1	5444368.49	17298082.39	17298082.39	23438060.67	17298082.39
2	5832505.31				
3	6021208.59				
4	6139978.28	18616491.73	18616491.73		18616491.73
5	6214731.83			18767908.19	
6	6261781.62				
7	6291394.75	18923192.26	18923192.26		25252340.04
8	6310033.23			18960945.29	
9	6321764.28				
10	6329147.78	6329147.78	6329147.78		
Total cost		**6137,18**	**6137,18**	**5850,75**	**5.867,4701**

Note that lot-sizes are determined based on the aggregated weekly requirements. One can easily found that the company can minimize cost through the appropriate selection of lot-sizing methods. According to the results, the efficiency of the heuristics of the WW increased compared to others. To present an overview of cost-saving, sensitivity analysis was conducted, and the results are presented in Figures 2a and 2b.

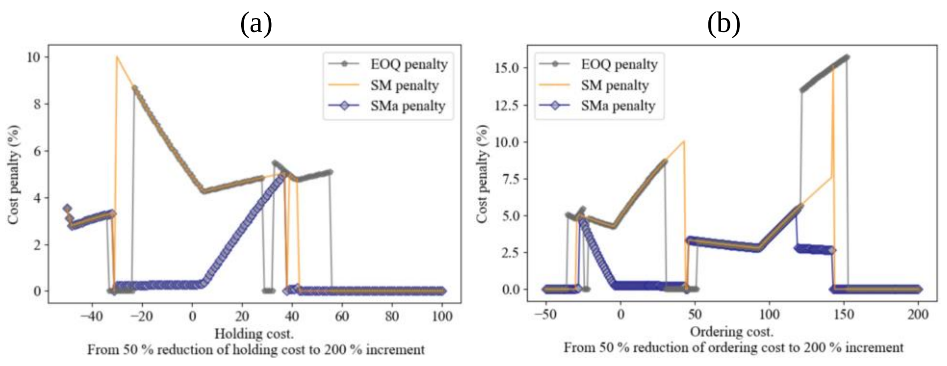


Fig. 2. Total cost comparison among lot-sizing methods EOQ, SM, MSM vs WW (10weeks horizon) from (a) 50% reduction of holding cost to 200% increment and (b) from 50% reduction of ordering cost to 200% increment

Figures 2a and 2b demonstrate that a company can save 0–16% cost through an appropriate replenishment decision. However, as reported by Hopp and Spearman (2011) no commercial MRP package actually uses WW algorithm. Therefore, this remains another dimension of the challenge faced by production managers.

Finally, this study focuses on the impact of aggregated planning horizon selection problem (Pedersen et al., 2020). Note that the factors contributing to the actual lot-sizing calculation and selection of optimization schemes mainly rely on: (1) the ratio between holding and ordering costs, and (2) demand pattern at the end of the horizon. Due to the end-effect, which exists due to the conversion from the T-period model horizon to the truncated n-period horizon, the second factor is important to include when comparing inventory lot-sizing performance (Van Den Heuvel & Wagelmans, 2005; Bach, Bocewicz, Banaszak & Muszyński, 2010). The proportional penalty of truncation is dependent on cost parameter also, even though the existence is a truism since some lot-sizing methods, e.g. SM Heuristic, are designed to cope with the situation of having a demand pattern continuing beyond the planning horizon. This means that "a replenishment is often scheduled unnecessarily close to the end of the horizon" (Silver and Miltenburg, 1984). As illustrated by above figures, the cost penalty of the inventory lot-sizing methods depends on both the holding and ordering cost, since the outcome of one would be the inverse function of the other; it is the ratio between the two costs that is of importance.

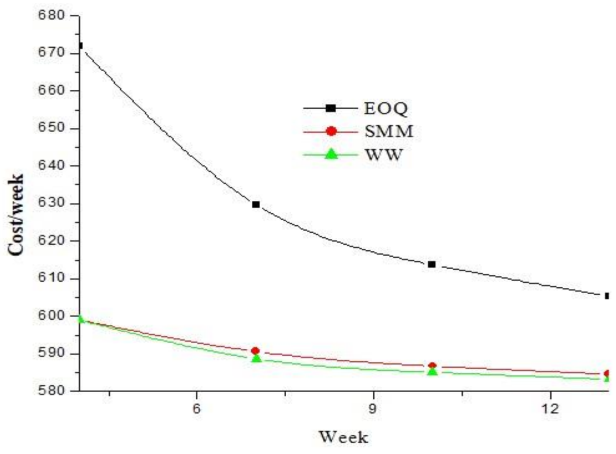


Fig. 3. Effect of lot-sizing methods with an integrated planning horizon

The graphical representation of cost per week when the aggregated planning horizons are considered as 4, 7, 10, and 13 weeks, respectively. Including all variations in the sensitivity analysis, the average penalty for implementation of the lot-sizing formula is nearly 17% , when only including the variations where an effect of the truncated horizon was existent, the average penalty of the SM heuristic was increased. It can be interpreted that the cost penalty in general increases, and becomes more diversified, between the lot sizing methods, as the ratio between ordering cost and holding cost increases when the holding cost decreases. When there is a difference between the penalty of SM heuristic and the

MSM heuristic, it is due to the penalty of the truncated horizon, since the MSM heuristic is performing optimally, underlining the importance of considering the effect. This is consistent with the conclusions of Kazan et al. (2000).

5. CONCLUSIONS

Integrated forecasting and inventory management have received considerable attention over several decades because of their implications for replenishment decision-making at both the strategic level and operational level for organizations. Although this pluralism is healthy from the perspective of knowledge advancement, one of the key issues faced by production managers in practice is how to integrate them? Moreover, the existing MRP or EPR system implementation is precisely defining the lot-sizing policy. Due to its computational complexity, the effect of the robust lot-sizing technique is ignored and a lot-for-lot policy is till practiced (Grubbström, Bogataj & Bogataj, 2010). On the other hand, statistical-forecasting techniques have also advanced significantly; however, they have not been used extensively at an operational level primarily due to their complexity (Syntetos et al., 2009). The dynamic lot-sizing problem behind our problem concerns related to a production plan that minimize total holding and ordering cost. Results demonstrates that the performance of EOQ or SM can deviate by 17% from the minimum cost obtained from the WW method; thus it is clear that WW dominates on both criteria.

Over the last few decades, the business environment of many industries has experienced great changes due to the integration of various frameworks for business analysis such as Collaborative Planning Forecasting and Replenishment (CPFR), Supply Chain Operations Reference-model (SCOR). However, researchers pointed out there are many hurdles both in-house and among the business partners that prevent or slow down business systems integration (Patalas-Maliszewska & Kłos, 2017; Alotaibi, 2016; Bocewicz, Nielsen & Banaszak, 2019). Indeed, approximately 66-70% of ERP implementation projects failed to accomplish their implementation objectives (Zabjek, Kovačič & Štemberger, 2009). In recent empirical research, Ali & Miller, 2017 also found that one of the most problematic and yet unresolved areas of ERP implementation is identifying and agreeing on the industry-standard implementation model. Moreover, as argued by Kourentzes et al. (2020), the inventory gains originate a more difficult optimization problem. In the context of MRP, Li & Disney (2017) also found that MRP systems are not always fully implemented, generate some consistency issues, and are unable to generate accurate data. This study took the initiative in this direction, and reported that although the company is tending towards the integration of modern business analytic, but struggling to achieve desirable performances. This is because the system they are trying to implement needs to be upgraded by introducing robust lot-sizing methods and train their work forces to adopt the process.

The investigation of performance of lot-sizing algorithms is always a classical topic in the production planning (Gola, 2014). This study has several limitations and can be extended in several directions. This study focused on a single-item setting, and purchase cost remains constant, there is no uncertainty in demand, and consequently the effect of safety stock. Therefore, an interesting area of future study is to consider multi-item production planning under demand uncertainty. Implementation of the presented framework for a rolling horizon would make the proposed framework closer to planning in practice, while it would also change the concerns of the truncated horizon effect, and its influence on the comparison between lot-sizing methods.

REFERENCES

Ali, M., & Miller, L. (2017). ERP system implementation in large enterprises–a systematic literature review. *Journal of Enterprise Information Management*, *30*(4), 666–692. https://doi.org/10.1108/JEIM-072014-0071

Alotaibi, Y. (2016). Business process modelling challenges and solutions: a literature review. *Journal of Intelligent Manufacturing*, *27*(4), 701–723. https://doi.org/10.1007/s10845-014-0917-4

Andriolo, A., Battini, D., Grubbström, R. W., Persona, A., & Sgarbossa, F. (2014). A century of evolution from Harris's basic lot size model: Survey and research agenda. *International Journal of Production Economics*, *155*, 16-38. https://doi.org/10.1016/j.ijpe.2014.01.013

Bach, I., Bocewicz, G., Banaszak, Z. A., & Muszyński, W. (2010). Knowledge based and CP-driven approach applied to multi product small-size production flow. *Control and Cybernetics*, *39*, 69–95.

Baker, K. R. (1989). Lot-sizing procedures and a standard data set: a reconciliation of the literature. *Journal of Manufacturing and Operations Management*, *2*(3), 199–221.

Bocewicz, G., Nielsen, P., & Banaszak, Z. (2019). Declarative modeling of a milk-run vehicle routing problem for split and merge supply streams scheduling. *Advances in Intelligent Systems and Computing*, *853*, 157–172. https://doi.org/10.1007/978-3-319-99996-8_15

Bocewicz, G., Nielsen, P., Banaszak, Z., & Thibbotuwawa, A. (2018). Routing and scheduling of unmanned aerial vehicles subject to cyclic production flow constraints. In *International Symposium on Distributed Computing and Artificial Intelligence* (pp. 75–86). Springer, Cham. https://doi.org/10.1007/978-3-319-99608-0_9

Box, G. E., Jenkins, G. M., & Reinsel, G. C. (2011). *Time series analysis: forecasting and control* (Vol. 734). John Wiley & Sons. https://doi.org/0.1111/jtsa.12194

De Bodt, M. A., Gelders, L. F., & Van Wassenhove, L. N. (1984). Lot sizing under dynamic demand conditions: A review. *Engineering Costs and Production Economics*, *8*(3), 165–187. https://doi.org/10.1016/0167188X(84)90035-1

Drexl, A., & Kimms, A. (1997). Lot sizing and scheduling—survey and extensions. *European Journal of operational research*, *99*(2), 221–235. https://doi.org/10.1016/S0377-2217(97)00030-1

Eriksen, P. S., & Nielsen, P. (2016). Order quantity distributions: Estimating an adequate aggregation horizon. *Management and Production Engineering Review*, *7*(3), 9–48. https://doi.org/10.1515/mper-2016-0024

Fildes, R., Goodwin, P., Lawrence, M., & Nikolopoulos, K. (2009). Effective forecasting and judgmental adjustments: an empirical evaluation and strategies for improvement in supplychain planning. *International journal of forecasting*, *25*(1), 3–23. https://doi.org/10.1016/j.ijforecast.2008.11.010

Gola A. (2014) Economic Aspects of Manufacturing Systems Design. *Actual Problems of Economics, 156*(6), 205–212.

Grubbström, R. W., & Tang, O. (2012). The space of solution alternatives in the optimal lotsizing problem for general assembly systems applying MRP theory. *International Journal of Production Economics*, *140*(2), 765777. https://doi.org/10.1016/j.ijpe.2011.01.012

Grubbström, R. W., Bogataj, M., & Bogataj, L. (2010). Optimal lotsizing within MRP theory. *Annual Reviews in Control*, *34*(1), 89–100. https://doi.org/10.3182/20090603-3-RU-2001.0562

Heady, R. B., & Zhu, Z. (1994). An improved implementation of the Wagner-Whitin Algorithm. *Production and Operations Management*, *3*(1), 55–63. https://doi.org/10.1111/j.1937-5956.1994.tb00109.x

Ho, C. J., & Ireland, T. C. (2012). Mitigating forecast errors by lot-sizing rules in ERP-controlled manufacturing systems. *International Journal of Production Research*, *50*(11), 3080–3094. https://doi.org/10.1080/00207543.2011.592156

Hopp, W. J., & Spearman, M. L. (2011). *Factory physics*. Waveland Press.

Kazan, O., Nagi, R., & Rump, C. M. (2000). New lot-sizing formulations for less nervous production schedules. *Computers & Operations Research*, *27*(13), 1325–1345. https://doi.org/10.1016/S0305-0548(99)00076-3

Kian, R., Berk, E., Gürler, Ü., Rezazadeh, H., & Yazdani, B. (2020). The effect of economies-of-scale on the performance of lot-sizing heuristics in rolling horizon basis. *International Journal of Production Research*, 1–15. https://doi.org/10.1080/00207543.2020.1730464

Kourentzes, N., Trapero, J. R., & Barrow, D. K. (2020). Optimising forecasting models for inventory planning. *International Journal of Production Economics*, *225*, 107597. https://doi.org/10.1016/j.ijpe.2019.107597

Li, Q., & Disney, S. M. (2017). Revisiting rescheduling: MRP nervousness and the bullwhip effect. *International Journal of Production Research*, *55*(7), 1992–2012. https://doi.org/10.1016/j.ijpe.2019.107597

Mills, T. C. (2019). *Applied Time Series Analysis: A Practical Guide to Modeling and Forecasting*. Academic Press.

Moon, I., Yoo, D. K., & Saha, S. (2016). The distribution-free newsboy problem with multiple discounts and upgrades. *Mathematical Problems in Engineering*, 2017253. https://doi.org/10.1155/2016/2017253

Nielsen, P., Jiang, L., Rytter, N. G. M., & Chen, G. (2014). An investigation of forecast horizon and observation fit's influence on an econometric rate forecast model in the liner shipping industry. *Maritime Policy & Management*, *41*(7), 667–682. https://doi.org/10.1080/03088839.2014.960499

Nilakantan, J. M., Li, Z., Tang, Q., & Nielsen, P. (2017). MILP models and metaheuristic for balancing and sequencing of mixed-model two-sided assembly lines. *European Journal of Industrial Engineering*, *11*(3), 353-379. https://doi.org/10.1504/EJIE.2017.084880

Patalas-Maliszewska, J. (2012). Assessing the Impact of Erp Implementation in the small Enterprises. *Foundations of management*, *4*(2), 51-62. https://doi.org/10.2478/fman-2013-0010

Patalas-Maliszewska, J., & Kłos, S. (2017). A Study on Improving the Effectiveness of a Manufacturing Company in the Context of Knowledge Management–Research Results. *Foundations of Management, 9*(1), 149160. https://doi.org/10.1515/fman-2017-0012

Pedersen, C. H., Nedbailo, V., Knudsen, M. G., Olesen, J., & Toft, S. (2020). *Analysis and development of an operations system*. P4 Semester Project, GBE4 gr. 16/2.016. Global Business Engineering, Aalborg University.

Saha, S., Das, S., & Basu, M. (2010). Optimal pricing and production lot-sizing for seasonal products over a finite horizon. *International Journal of Mathematics in Operational Research*, *2*(5), 540–553. https://doi.org/10.1504/IJMOR.2010.03434

Silver, E. A., & Meal, H. C. (1973). A heuristic for selecting lot size quantities for the case of a deterministic time-varying demand rate and discrete opportunities for replenishment. *Production and Inventory Management*, *2*, 64–74.

Silver, E. A., Pyke, D. F., & Thomas, D. J. (2016). *Inventory and production management in supply chains*. CRC Press.

Silver, E., & Miltenburg, J. (1984). Two modifications of the SilverMeal lot sizing heuristic. *INFOR: Information Systems and Operational Research*, *22*(1), 56–69. https://doi.org/10.1080/03155986.1984.11731912

Świć, A., & Gola, A. (2013). Economic Analysis of Casing Parts Production in a Flexible Manufacturing System. *Actual Problems of Economics*, *141*(3), 526–533.

Syntetos, A. A., Boylan, J. E., & Disney, S. M. (2009). Forecasting for inventory planning: a 50-year review. *Journal of the Operational Research Society*, *60,* 149–S160. https://doi.org/10.1057/jors.2008.173

Syntetos, A. A., Nikolopoulos, K., & Boylan, J. E. (2010). Judging the judges through accuracy-implication metrics: The case of inventory forecasting. *International Journal of Forecasting*, *26*(1), 134-143. https://doi.org/10.1016/j.ijforecast.2009.05.016

Taneja, K., Ahmad, S., Ahmad, K., & Attri, S. D. (2016). Time series analysis of aerosol optical depth over New Delhi using Box–Jenkins ARIMA modeling approach. *Atmospheric Pollution Research*, *7*(4), 585596. https://doi.org/10.1016/j.apr.2016.02.004

Van Den Heuvel, W., & Wagelmans, A. P. (2005). A comparison of methods for lot-sizing in a rolling horizon environment. *Operations Research Letters*, *33*(5), 486–496. https://doi.org/10.1016/j.orl.2004.10.001

Wagner, H. M., & Whitin, T. M. (1958). Dynamic version of the economic lot size model. *Management science*, *5*(1), 89-96. https://doi.org/10.1287/mnsc.5.1.89

Xi, M. H., Wang, H. X., & Zhao, Q. H. (2012). Regression Based Integration of Demand Forecasting and Inventory Decision. *Advanced Materials Research, 433,* 2954–2956. https://doi.org/10.4028/www.scientific.net/AMR.433-440.2954

Zabjek, D., Kovačič, A., & Štemberger, M. I. (2009). The influence of business process management and some other CSFs on successful ERP implementation. *Business Process Management Journal*, *15*(4), 588–608. https://doi.org/10.1108/14637150910975552

Applied Computer Science, vol. 16, no. 4, pp. 37–47
doi:10.23743/acs-2020-27

Submitted: 2020-11-28
Revised: 2020-12-04
Accepted: 2020-12-11

ECG, software, algorithm testing, heart

Marcin MACIEJEWSKI [0000-0001-9116-5481]*,
Barbara MACIEJEWSKA[0000-0001-6797-7519]**,
Robert KARPIŃSKI[0000-0003-4063-8503]***,
Przemysław KRAKOWSKI [0000-0001-7137-7145]****

ELECTROCARDIOGRAM GENERATION SOFTWARE FOR TESTING OF PARAMETER EXTRACTION ALGORITHMS

Abstract

Fast and automated ECG diagnosis is of great benefit for treatment of cardiovascular and other conditions. The algorithms used to extract parameters need to be precise, robust and efficient. Appropriate training and testing methods for such algorithms need to be implemented for optimal results. This paper presents a software solution for computer ECG generation and a simplified concept of testing process. All the parameters of the resulting generated signal can be tweaked and set properly. Such software can also be beneficial for training and educational use.

1. INTRODUCTION

Machine – supported diagnosis has been a hot topic ever since first computers became available. This trend has become more noticeable with the advent of IoT and interconnectivity (Rincón et al., 2009). It has become even more relevant now due to the pandemic situation preventing patients from direct interaction with

[*] Lublin University of Technology, Faculty of Electrical Engineering and Computer Science, Institute of Electronics and Information Technology, Nadbystrzycka 36, 20-618 Lublin, Poland, m.maciejewski@pollub.pl
[**] Independent researcher, Lublin, Poland, barbarasmaciejewska@gmail.com
[***] Lublin University of Technology, Faculty of Mechanical Engineering, Department of Machine Design and Mechatronics, Nadbystrzycka 36, 20-618 Lublin, Poland, r.karpinski@pollub.pl
[****] Medical University of Lublin, Chair and Department of Traumatology and Emergency Medicine, Staszica 11, 20-081 Lublin, Poland, przemyslawkrakowski@umlub.pl

medical staff. Introduction of widespread online diagnostics systems for various medical conditions seems to be inevitable, and these will become a key element in future healthcare systems (Surtel, Maciejewski & Maciejewska, 2013). However, local patient monitoring systems are not without their own limitations. Not all procedures are safe to be performed remotely by the patients or their family due to its invasiveness. For safety reasons, it is recommended to use mostly non – invasive devices to measure parameters potentially crucial to patient condition assessment (Karpiński, Machrowska & Maciejewski, 2019; Machrowska, Karpiński, Krakowski & Jonak, 2019; Maciejewski et al., 2014).

Electrocardiography, is one of the most commonly used method of patient examination. The procedure is simple and non – invasive, as it uses surface electrodes which need to be placed on the skin in certain points. The weak voltage on the skin surface is amplified and can be viewed on a scope directly, or sampled by and A/D converter. The signal is then processed to obtain valuable information about circulatory system. Such information is vital both in routine assessment as well as in acute, life-threatening conditions. ECG can be performed with the use of simple, cheap and light devices at home, in the field or in smaller clinics, or more advanced devices with additional functionality in hospitals and various care centers.

2. ECG BASICS

Heart is an organ composed of muscle tissue and is located to the left of the sternum. It can be divided into left and right ventricle and left and right atrium. These chambers are separated by valves. In healthy individuals an electric impulse is generated in the sinoaortal node, than it travels to the atrioventricular node and to two bundles of His. This conductive cycle causes atrial contractions, followed by ventricle contraction and depolarization (Reisner, Clifford & Mark, 2006). Blood is being pumped to the lungs and the rest of the body. The electrical change can be visualized by a vector of electric potential that changes in time. It is possible to measure the electrical activity of the heart using specialized low voltage amplifiers. The resulting signal carries significant information that can be used to diagnose various heart conditions. It can be divided into waves, segments, intervals and complexes (Waechter, 2012).

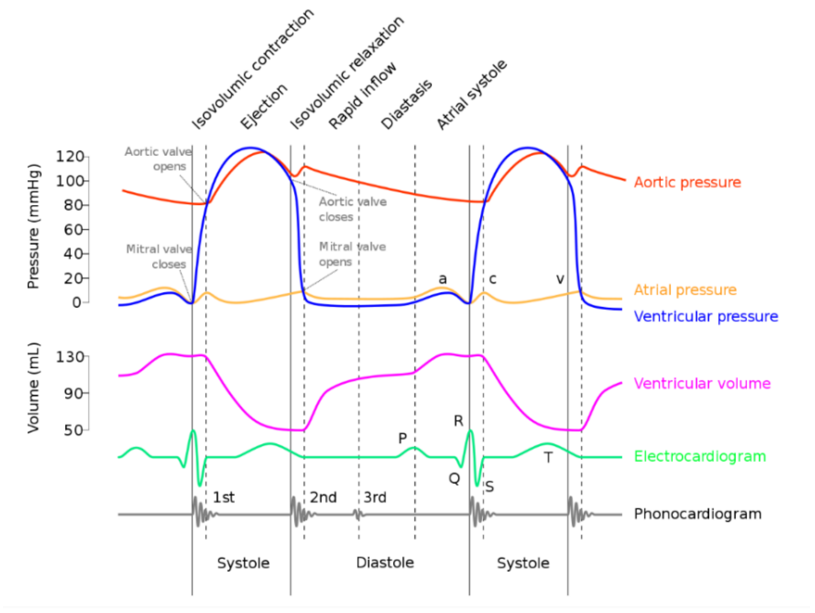


Fig. 1. Wiggers diagram – the diagram shows the aortic, atrial and ventricular pressure and the ventricular volume in relation to the ECG signal

The figure 1 (Xavax, 2016) presents changes in parameters during heart cycles. During diastole aortic pressure decreases and blood flows into the ventricles. When electrical activity forces the muscles to contract the ventricular pressure increases and blood is pumped from the heart to the system.

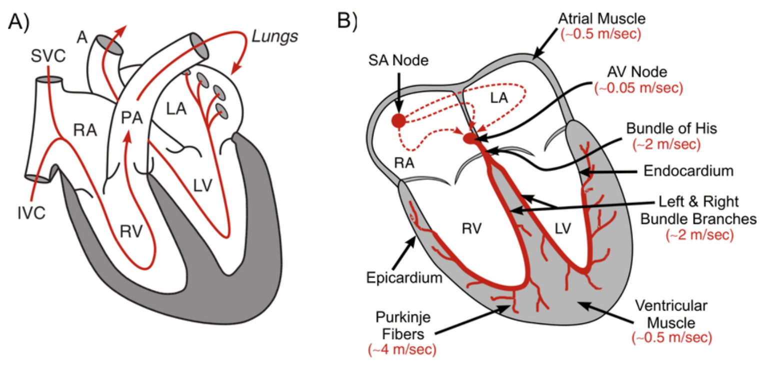


Fig. 2. Blood flow(A) and electrical activity (B) of the heart

Figure 2 (Costa, 2016) presents the direction of blood flow and electrical activity of the heart. Electrical pulse is generated in the sinoatrial node. The potential wave then travels to the rest of the heart causing contraction of muscles.

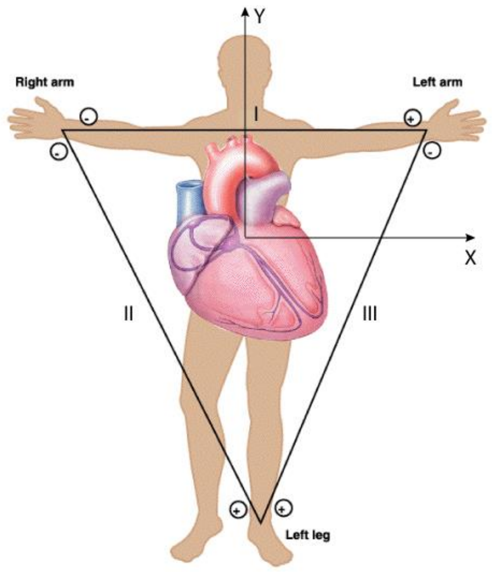


Fig. 3. Electrode placement during three lead ECG according to the Einthoven triangle (Burhan, 2011)

The figure above presents basic method for ECG electrode placement. The electrodes can be placed on arms and left leg or on torso and abdomen. Improper placement, inadequate skin preparation and damaged cabling are frequent causes of errors during procedure. These errors can result in significant artifacts or even make the signal unusable.

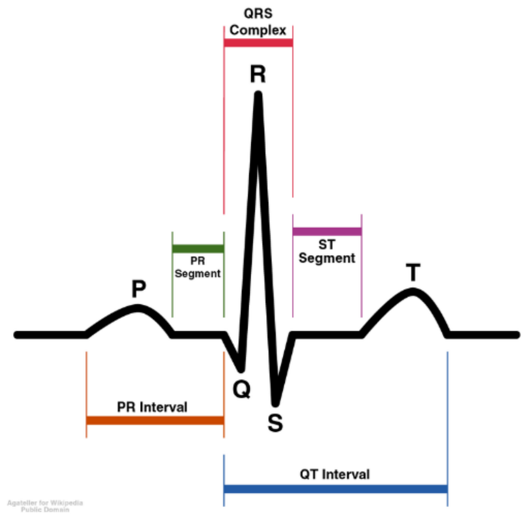


Fig. 4. Time parameters of the ECG signal commonly used for diagnosis

During ECG signal processing it is important to extract the following parameters (Luthra, 2007):

- presence of P and T waves and QRS complex in the proper order
- position, duration and amplitude of P and T waves and QRS complex,
- duration of the PR and QT intervals,
- duration of the PR and ST segments,
- length of the heart cycle, commonly calculated as time before consecutive QRS complexes,
- variability of the previously mentioned parameters.

Abnormalities in values of these parameters can be directly tied to various conditions, like arrythmias, fluttering, branch blocks, abnormal heart positioning etc. For example, changes can be caused by electrolytic imbalance, drugs, hyper- and hypothermia, anxiety and overall nervous system condition.

3. ECG DIAGNOSIS AND CLASSIFICATION

Classical ECG diagnostics is based on expert knowledge and manual signal analysis (Waechter, 2012). It is prone to human errors and the results are often presented using descriptive language, like "slightly raised", "delayed" etc., which makes it difficult for later use the data in comparative study (Maciejewski, 2019). Nowadays, it is a common practice to include computer aided diagnosis methods into the decision process (Omiotek, 2017; Rehman, Mustafa & Israr, 2013). By basing the decision on a large set of predetermined rules and cases in the database it is possible to positively influence the overall process by helping or hinting the specialist (Omiotek, Dzierżak & Uhlig, 2019).

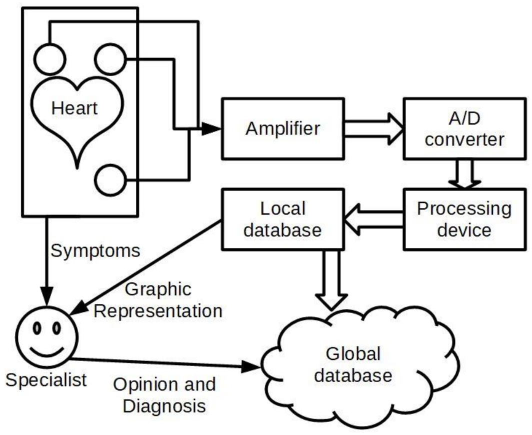


Fig. 5. A simplified overview of the ECG diagnosis process

Computer aided ECG diagnosis is based on a set of steps. These include:

- data acquisition using a sensor or a bank of sensors,
- signal preprocessing including filtration and denoising,
- feature extraction using chosen algorithms,
- classification and comparison with available knowledge base,
- presenting diagnosis.

Each step is complex in it is own way (Boulakia, Cazeau, Fernández, Gerbeau & Zemzemi, 2010). Various errors can be made during acquisition due to improper electrode placement, electronical malfunctions and data recording (Barill & SlikkStat Learning Inc., 2012). Filtration and denoising can be challenging when significant biological or technical artifacts are present in the signal (Rehman, Mustafa & Israr, 2013). Due to these factors it is sometimes hard to extract features and perform classification. These artifacts include signal drift, electrical activity of skeletal and respiratory muscles, changes in voltage due to varying skin impedance, power line interference, quantization noise etc (Bronzino, 2000; Clifford, Azuaje & Mcsharry, 2006). Filtration can be performed using, classical filters and FFT or wavelet methods (Ławicki & Zhirnova, 2015). During this process certain frequencies like 50 or 60Hz, which correspond with power line interference, need to be filtered out (Bronzino, 2000), although improper filter selection can lead to loss of sharp inflections needed to adequately process the QRS complex.

Feature extraction is usually performed in steps (Zhou, Hou & Zuo, 2009). Firstly, the position of the QRS complex is determined using various methods (Bronzino, 2000; Pan & Tompkins, 1985). Afterwards, positions of P and T waves are calculated. Lastly, durations and time intervals are estimated. Especially the last step can be hard, as the waves often are lacking a significant detectable inflection point in the beginning and end of the wave. After that it is necessary to determine parameter variability between consecutive heart cycles. In some cases when the condition presents itself randomly and it's occurrence is separated by long intervals of normal heart function it is necessary to perform long term analysis. Various acute heart conditions result in significant electrical activity variations, which in turn presents a real challenge for the designers of algorithms for ECG processing. Errors in QRS position detection are the most meaningful, as it is often used as a reference point in further analysis (Maciejewski & Dzida, 2017). Typical approaches use:

- filter banks,
- thresholding,
- polynomial estimation,
- frequency analysis,
- state machines,
- mixture of the above.

More sophisticated and complex software can combine several approaches or determine the best approach for each individual case.

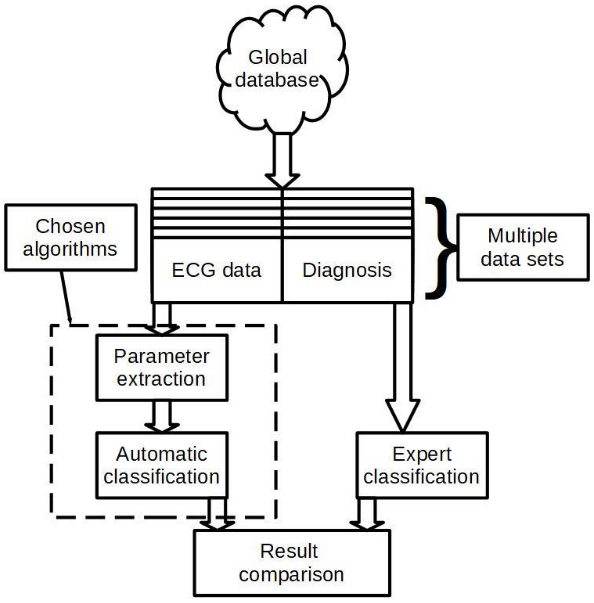


Fig. 6. Computer aided diagnosis process.

4. ECG GENERATION SOFTWARE

As previously shown, ECG analysis is a complex process. Multiple factors can result in improper results. It is possible to access vast, free databases containing ECG signals obtained from both healthy individuals and patients suffering from various cardiacconditions. This data usually has been processed and analyzed by a specialist, and is accompanied by diagnostic data. All mentioned above facts account for simplicity of ECG processing algorithms designing and testing. This process, however, can be improved.

Let’s consider a situation, when one has to determine robustness of a method while processing a noisy signal. It is necessary to check the method’s performance for multiple SNR(Signal to Noise Ratio). One has to perform these steps:

1. Choose a set of data as a basis for algorithm testing.
2. Extract ECG parameters and use them as reference points.
3. Apply noise to the signal.
4. Extract parameters and compare them with reference points.
5. Repeat until all values of SNR are tested.

A lot depends on the first two steps. It is not possible to compile a data set spanning all probable cases with varying heart rate or amplitudes and lengths of waves and QRS complex. Creating robust and accurate extraction algorithms requires testing them in as many cases as possible. This is where ECG generation software comes in.

The purpose of this type of application is generating ECG signals on demand based on input parameters. In this example a Java application was created. The software has the following functionalities:

1. Reading from an input file containing a set of parameters for further ECG generation. These include amplitudes and lengths of waves and lengths of pauses between them. Additionally, it is possible to include a randomness factor for each parameter. It is possible to generate signals corresponding to a chosen number of heart cycles.
2. Generating the signal based on the input file. The signal is constructed from exponential functions and linear segments.
3. Generating a .csv file containing information about the values of parameters for every heart cycle separately. It is necessary to save this information when randomness factor was introduced. This is the main advantage of this proposed approach.

Having access to information corresponding to each heart cycle means, that it is possible to directly compare the results from the ECG processing algorithm under test with actual values. Generating very long signals with multiple heart cycles and introducing randomness results in large sets of input data. It is also easy to automate robustness tests by introducing additional noise of chosen type and amplitude.

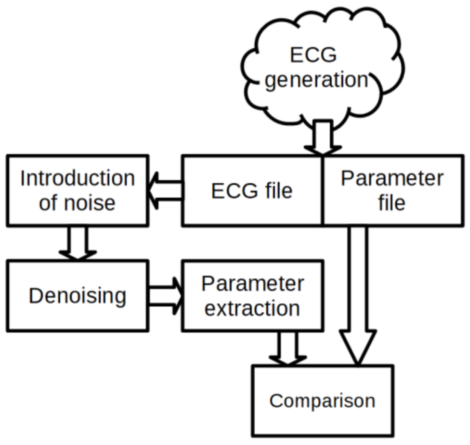


Fig. 7. Testing parameter extraction methods using ECG generation software

Additionally, it is possible to prepare a set of input parameters for various heart conditions like tachycardia, bradycardia etc. The tool is being used in the teaching process of biomedical engineering students at Lublin Institute of Technology, Poland.

Sample signals generated by the software are presented in the figures below, with additional markers corresponding to input parameters.

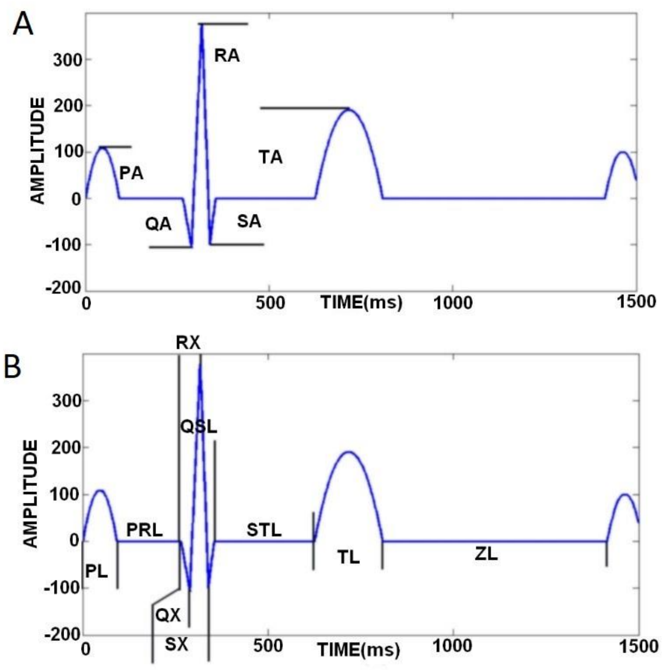


Fig. 8. Input parameters used by the software – both amplitude parameters (A) and time parameters (B) are used

5. FUTURE RESEARCH AND DEVELOPMENT

The software presented in this paper can be improved in many ways. Firstly, introduction of a proper GUI with parameter input windows and on-line heart cycle visualization will make it more intuitive to use by students and staff. Simpler saving and loading of parameter files will be introduced. Furthermore, an inbuilt database of various sets of parameters corresponding to heart conditions will be designed and implemented. Afterwards, methods for introduction of noise and artifacts will be included. Lastly, this software may become available to download in Java version or Python after porting.

REFERENCES

Barill, T., & SlikkStat Learning Inc. (2012). *The six second ECG: A practical guide to basic and 12 lead ECG interpretation*. Palm Springs, Calif.: SkillStat Learning Inc.

Boulakia, M., Cazeau, S., Fernández, M. A., Gerbeau, J.-F., & Zemzemi, N. (2010). Mathematical Modeling of Electrocardiograms: A Numerical Study. *Annals of Biomedical Engineering*, *38*(3), 1071–1097. https://doi.org/10.1007/s10439-009-9873-0

Bronzino, J. D. (2000). *The biomedical engineering handbook*. Boca Raton, Fla.: CRC Press in cooperation with IEEE Press.

Burhan, A. (2011). Einthoven triangle ECG. Retrieved 19 December 2020, from Medicalopedia website: https://mk0medicalopediwjftu.kinstacdn.com/wp-content/uploads/2011/11/einthoven-triangle-ecg.jpg

Clifford, G. D., Azuaje, F., & Mcsharry, P. (2006). ECG statistics, noise, artifacts, and missing data. *Advanced Methods and Tools for ECG Data Analysis*, *6*, 18.

Costa, C. M. (2016). *Computational Modeling of Bioelectrical Activity of the Heart at Microscopic and Macroscopic Size Scales* (Doctoral dissertation). Karl-Franzens Universit ̈at Graz, Graz. https://doi.org/10.13140/RG.2.2.26259.99365

Karpiński, R., Machrowska, A., & Maciejewski, M. (2019). Application of acoustic signal processing methods in detecting differences between open and closed kinematic chain movement for the knee joint. *Applied Computer Science, 15*(1), 36–48. https://doi.org/10.23743/acs-2019-03

Ławicki, T., & Zhirnova, O. (2015). Application of curvelet transform for denoising of CT images. In *Photonics Applications in Astronomy, Communications, Industry, and High-Energy Physics Experiments 2015*, (966226). International Society for Optics and Photonics. https://doi.org/10.1117/12.2205483

Luthra, A. (2007). *ECG made easy*. New Delhi; Tunbridge Wells: Jaypee ; Anshan Ltd.

Machrowska, A., Karpiński, R., Krakowski, P., & Jonak, J. (2019). Diagnostic factors for opened and closed kinematic chain of vibroarthrography signals. *Applied Computer Science, 15*(3), 34-44. http://doi.org/10.23743/acs-2019-19

Maciejewski, M. (2019). Information technology implementations and limitations in medical research. *Informatyka, Automatyka, Pomiary w Gospodarce i Ochronie Środowiska*, *5*(1), 66–72. https://doi.org/10.5604/20830157.1148052

Maciejewski, M., & Dzida, G. (2017). ECG parameter extraction and classification in noisy signals. *2017 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA)* (pp. 243–248). IEEE. https://doi.org/10.23919/SPA.2017.8166872

Maciejewski, M., Surtel, W., Wójcik, W., Masiak, J., Dzida, G., & Horoch, A. (2014). Telemedical systems for home monitoring of patients with chronic conditions in rural environment. *Ann Agric Environ Med., 21*(1), 167-73.

Omiotek, Z. (2017). Improvement of the classification quality in detection of Hashimoto's disease with a combined classifier approach. *Proceedings of the Institution of Mechanical Engineers, Part H: Journal of Engineering in Medicine*, *231*(8), 774–782.

Omiotek, Z., Dzierżak, R., & Uhlig, S. (2019). Fractal analysis of the computed tomography images of vertebrae on the thoraco-lumbar region in diagnosing osteoporotic bone damage. *Proceedings of the Institution of Mechanical Engineers, Part H: Journal of Engineering in Medicine*, *233*(12), 1269–1281.

Pan, J., & Tompkins, W. J. (1985). A Real-Time QRS Detection Algorithm. *IEEE Transactions on Biomedical Engineering, BME-32*(3), 230–236. https://doi.org/10.1109/TBME.1985.325532

Rehman, A., Mustafa, M., & Israr, I. (2013). Survey of wearable sensors with comparative study of noise reduction ecg filters. *International Journal of Computing and Network Technology*, *221*(1249), 1–21.

Reisner, A., Clifford, G., & Mark, R. (2006). *The Physiological Basis of the Electrocardiogram*.

Rincón, F. J., Gutiérrez, L., Jiménez, M., Díaz, V., Khaled, N., Atienza, D., … Micheli, G. D. (2009). Implementation of an Automated ECG-based Diagnosis Algorithm for a Wireless Body Sensor Plataform. *Proceedings of the International Conference on Biomedical Electronics and Devices (BIODEVICES 2009)* (pp. 88–96). Porto, Springer.

Surtel, W., Maciejewski, M., & Maciejewska, B. (2013). Processing of simultaneous biomedical signal data in circulatory system conditions diagnosis using mobile sensors during patient activity. *2013 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA)* (pp. 163–167). IEEE.

Waechter, J. (2012). *Introduction to ECG's: Rhythm Analysis*. Jason Waechter.

Xavax. (2016). *A Wiggers diagram, showing the cardiac cycle events occuring in the left ventricle.* Wikimedia Commons: Wiggers Diagram.svg. Retrieved from https://commons.wikimedia.org/w/index.php?curid=50317988

Zhou, H., Hou, K.-M., & Zuo, D. (2009). Real-Time Automatic ECG Diagnosis Method Dedicated to Pervasive Cardiac Care. *Wireless Sensor Network*, *01*(04), 276–283. https://doi.org/10.4236/wsn.2009.14034

Applied Computer Science, vol. 16, no. 4, pp. 48–55
doi:10.23743/acs-2020-28

Submitted: 2020-10-17
Revised: 2020-12-02
Accepted: 2020-12-17

architectural paradigm, pattern, module, cloud technology

Denis RATOV [0000-0003-4326-3030]*

ARCHITECTURAL PARADIGM OF THE INTERACTIVE INTERFACE MODULE IN THE CLOUD TECHNOLOGY MODEL

Abstract

The article discusses an architectural template for building a module for organizing the work of a multiuser windowed information web-system. To solve this problem, JavaScript objects have been created: a window manager object and a window interactive interface class, which allow a web application to function when organizing cloud technologies. The software implementation is considered and the results of the practical use of the developed module are presented.

1. INTRODUCTION

Today, when developing information systems, cloud technologies are often used for remote computing and data processing (Medvedev, 2013). Cloud computing refers to the provision of computer resources and capacities to the user in the form of Internet services. Cloud computing is a distributed data processing process in which computer resources and network capacity are provided to the user as an Internet service (Papadopoulos & Katsaros, 2011). Cloud technology inherently implements the processes of creating cloud applications and organizes work with them, without the introduction of additional software. Typically, for such applications, functionality is created in a web browser environment. Such a software product is a client-server application with a Web interface that provides the user with the ability to access data from any

* Volodymyr Dahl East Ukrainian University, Faculty of Information Technology and Electronics, Department of Programming and Mathematics, Tsentralnyi Ave, 59A Severodonetsk, Luhansk Oblast, Ukraine, 93400, denis831102@gmail.com

active point, provided that they are connected to the Internet. Cloud data processing or computing is not provided on the clients' personal computers, but on powerful server computers. For effective interaction of the client with remote data without completely reloading the current page, consider the user interface template, which is put into the structure of a module that implements controls, input, sending and receiving data in the form of windowed web forms with their inherent functionality in the browser context.

By a web form we mean an independent fragment of the user interface with its own logic of behavior, for the display of which the template objects of the module being developed are used. One of the purposes of such a module is to reuse it. This allows you to define the functionality of objects once and use them in different contexts and information systems.

When developing the module solves the following problem: you need a reliable encapsulated namespace in which you can define the data and functionality of objects. This makes it possible to make some of this data available and to limit the functionality of others.

Today there are web technologies, and libraries and frameworks developed on their basis for creating web applications and user interfaces designed for information systems to work in browsers. The processes of standardization of HTML (HTML 4.01 Specification, n.d.), CSS (Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification, n.d.) and JavaScript (ECMAScript Language Specification – ECMA-262 Edition 5.1, n.d.) languages allowed to achieve not only a high degree of cross-platform user interfaces, but also a fairly good degree of cross-browser compatibility, so the use of appropriate standards when building Web applications has become the dominant approach.

When developing modular information systems, standards alone are not enough: design patterns, libraries of standard controls, support for presentation logic (for example, Presenter in the MVP model (MVP architecture, n.d.)) and much more are needed. The corresponding tools are still in their infancy. Examples of free products are (MediaWiki, n.d.), Drupal (Drupal – Open Source CMS, n.d.), WordPress. The inevitable payback for such systems is the binding to server technologies, which limits their application in situations where the server environment is fixed for the developer. When considering client libraries that do not depend on server technologies, their specialization is visible: on manipulating the DOM model (jQuery (jQuery, n.d.), Zepto.js (Zepto.js: the aerogel-weight jQuery-compatible JavaScript library, n.d.)), styling pages and controls (Bootstrap, n.d.; jQuery UI, n.d.; w2ui (w2ui: Home, n.d.)), building application frameworks (AngularJS (AngularJS – Superheroic JavaScript MVW Framework, n.d.), Backbone.js (Backbone.js, n.d.), Knockout (Knockout: Home, n.d.)).

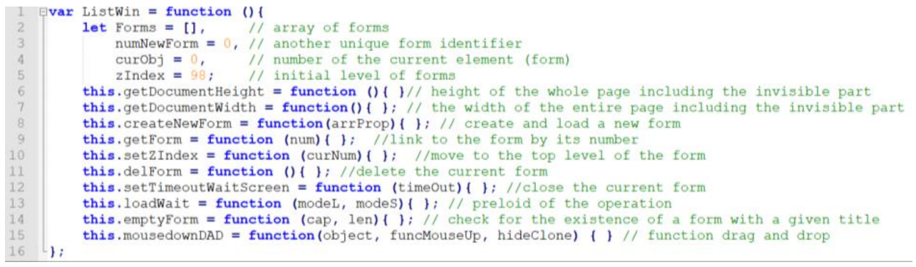
Despite the rich set among the existing tools, there are tasks that are relevant in the development of information systems: the presence of a dispatcher of interface models according to a given specification, data integrity control with the possibility of multi-user access.

2. MODULE ARCHITECTURE

The module being developed consists of two relatively independent, interacting with each other components, which are implemented as JavaScript objects: a window manager object and a window interactive interface class. Let's use JavaScript's mechanism for accessing objects using the new operator, which is used to create objects using the function of our own constructor, thereby creating an analogue of the class. Such a constructor stores an instance of the object in the closure. This prevents changes to the object outside of the constructor function. This uses an object creation template called a "module" and an isolated namespace template (Stoyan Stefanov, 2011). In order to change not only the appearance of the displayed form components, but also their behavior without modifying the main objects of the module, a design principle called the template method was applied during development (Gamma, Helm, Johnson & Vlissides, 2001). The library code contains a method for setting callback functions where the developer needs to supplement the standard data and event handling with his own actions.

The mechanism of the user interactive interface, in addition to the functions of working with interface components, must meet the following requirements:

1. The web form object must have a method for sending an ajax request to the server and be able to process server responses.
2. Web form object can be either a simple set of fields or contain subsections, tabs or tables.
3. The appearance of web forms must be customizable.
4. There can be several web forms on the page that can interact with each other.



```
var ListWin = function (){
    let Forms = [],      // array of forms
        numNewForm = 0, // another unique form identifier
        curObj = 0,      // number of the current element (form)
        zIndex = 98;     // initial level of forms
    this.getDocumentHeight = function (){ }// height of the whole page including the invisible part
    this.getDocumentWidth = function(){ }; // the width of the entire page including the invisible part
    this.createNewForm = function(arrProp){ }; // create and load a new form
    this.getForm = function (num){ };  //link to the form by its number
    this.setZIndex = function (curNum){ };  //move to the top level of the form
    this.delForm = function (){ }; //delete the current form
    this.setTimeoutWaitScreen = function (timeOut){ }; //close the current form
    this.loadWait = function (modeL, modeS){ }; // preloid of the operation
    this.emptyForm = function (cap, len){ }; // check for the existence of a form with a given title
    this.mousedownDAD = function(object, funcMouseUp, hideClone) { } // function drag and drop
};
```

Fig. 1. Dispatcher object architecture

The developed window manager (ListWin) is a JavaScript object that:

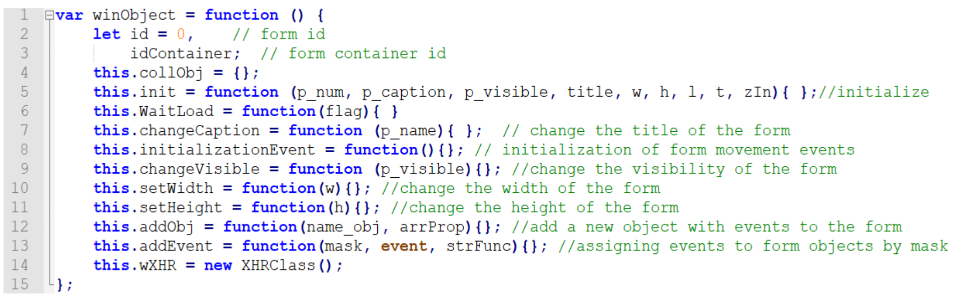
1. Stores a collection of generated web forms with their components.
2. Provides a high-level API for manipulating web forms and data from client controls.
3. Provides interaction of the web form with the DOM model of the web document.
4. Performs preliminary visualization of running processes on forms.
5. Provides a drag and drop mechanism for controls.

The ListWin dispatcher implementation consists of its own constructor with privat fields and public methods. Figure 1 shows the object architecture of the dispatcher.

The JavaScript object of the windowed interactive interface is responsible for the operation of the application, handling the events of the form component and includes:

1. Methods for rendering and manipulating the web form: init(), changeCaption(), changeVisible(), setWidth(), WaitLoad(), addObj().
2. Web form event trigger constructor: initializationEvent().
3. Constructor of event handlers for web form controls: addEvent().

Let's consider the implementation of the window interface object (Fig. 2). The functions of an object are implemented as its public methods.



```
var winObject = function () {
    let id = 0,     // form id
        idContainer;  // form container id
    this.collObj = {};
    this.init = function (p_num, p_caption, p_visible, title, w, h, l, t, zIn){ };//initialize
    this.WaitLoad = function(flag){ }
    this.changeCaption = function (p_name){ };  // change the title of the form
    this.initializationEvent = function(){}; // initialization of form movement events
    this.changeVisible = function (p_visible){}; //change the visibility of the form
    this.setWidth = function(w){}; //change the width of the form
    this.setHeight = function(h){}; //change the height of the form
    this.addObj = function(name_obj, arrProp){}; //add a new object with events to the form
    this.addEvent = function(mask, event, strFunc){}; //assigning events to form objects by mask
    this.wXHR = new XHRClass();
};
```

Fig. 2. Implementing the window interface object

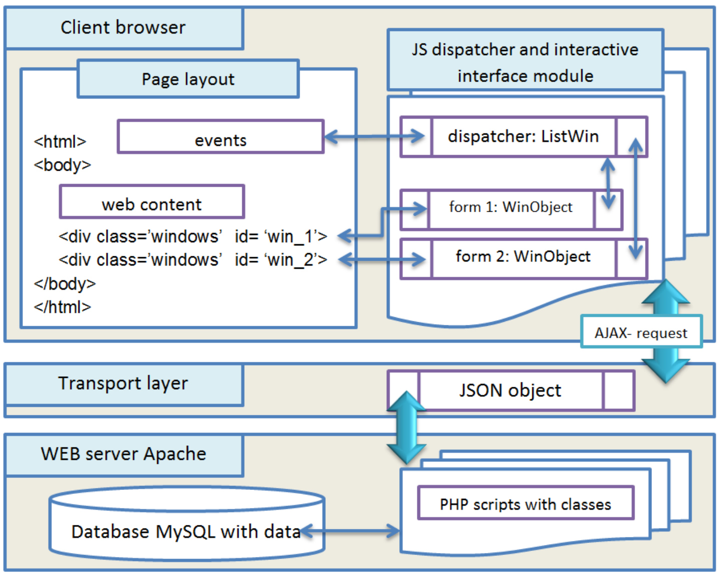


Fig. 3. Module components interaction scheme

Using an object XHRClass the transport layer of interaction between the client and the server is implemented, namely, loading data for web forms from the server, saving data to the server, asynchronous AJAX requests to methods of php objects on the server. All data transfer takes place in the background without reloading the page (Crane & Pascarello, 2006). The JSON format is used to transfer structured data between client and server.

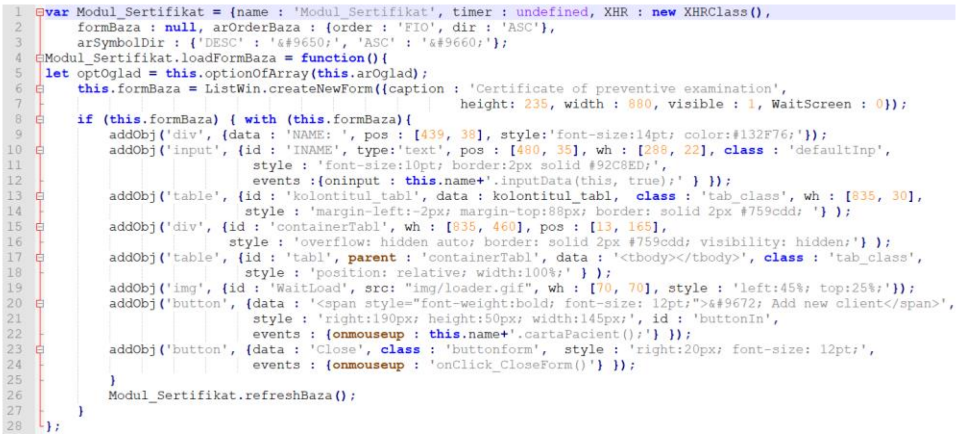
The interaction scheme of the module components is shown in Fig. 3.

When constructing a separate application module based on the described objects of the dispatcher and the interactive window interface, the basic principle of creating objects can be a structural-hierarchical relationship. This approach allows you to design individual interactive interfaces in the form of application modules.

At the level of interaction of web-forms with each other, structural approaches are no longer effective enough, since only a small number of forms are in fixed master-subordinate relationships. Therefore, at this level, a transition was made to the network model of the organization (Fig. 3): forms win_1, win_2 are independent acting objects that react to the events of their components using callback functions assigned during construction by the ListWin dispatcher.

3. MODULE IMPLEMENTATION RESULTS

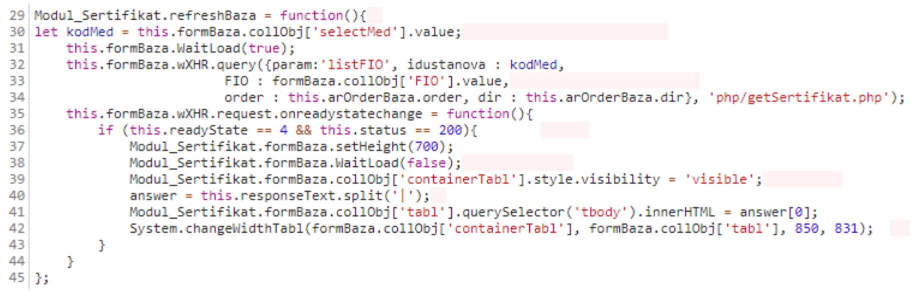
Consider this model and an architectural template using the example of implementing the module for creating certificates Modul_Sertifikat (Fig. 4). Using the loadFormBaza() method, the object provides manipulations with the DOM model of the web document. This uses the createNewForm() constructor of the ListWin dispatcher and its addObj() method to add controls with event handlers.



```
var Modul_Sertifikat = {name : 'Modul_Sertifikat', timer : undefined, XHR : new XHRClass(),
     formBaza : null, arOrderBaza : {order : 'FIO', dir : 'ASC'},
     arSymbolDir : {'DESC' : '&#9650;', 'ASC' : '&#9660;'}};
Modul_Sertifikat.loadFormBaza = function(){
let optOglad = this.optionOfArray(this.arOglad);
     this.formBaza = ListWin.createNewForm({caption : 'Certificate of preventive examination',
                                             height: 235, width : 880, visible : 1, WaitScreen : 0});
     if (this.formBaza) { with (this.formBaza){
          addObj('div', {data : 'NAME: ', pos : [439, 38], style:'font-size:14pt; color:#132F76;'});
          addObj('input', {id : 'INAME', type:'text', pos : [480, 35], wh : [288, 22], class : 'defaultInp',
                           style : 'font-size:10pt; border:2px solid #92C8ED;',
                           events :{oninput : this.name+'.inputData(this, true);' } });
          addObj('table', {id : 'kolontitul_tabl', data : kolontitul_tabl,  class : 'tab_class', wh : [835, 30],
                          style : 'margin-left:-2px; margin-top:88px; border: solid 2px #759cdd; '} );
          addObj('div', {id : 'containerTabl', wh : [835, 460], pos : [13, 165],
                        style : 'overflow: hidden auto; border: solid 2px #759cdd; visibility: hidden;'} );
          addObj('table', {id : 'tabl', parent : 'containerTabl', data : '<tbody></tbody>', class : 'tab_class',
                          style : 'position: relative; width:100%;' } );
          addObj('img', {id : 'WaitLoad', src: "img/loader.gif", wh : [70, 70], style : 'left:45%; top:25%;'});
          addObj('button', {data : '<span style="font-weight:bold; font-size: 12pt;">&#9672; Add new client</span>',
                           style : 'right:190px; height:50px; width:145px;', id : 'buttonIn',
                           events : {onmouseup : this.name+'.cartaPacient();'} });
          addObj('button', {data : 'Close', class : 'buttonform',  style : 'right:20px; font-size: 12pt;',
                           events : {onmouseup : 'onClick_CloseForm()'} });
          }
          Modul_Sertifikat.refreshBaza();
     }
};
```

Fig. 4. Certificate creation module

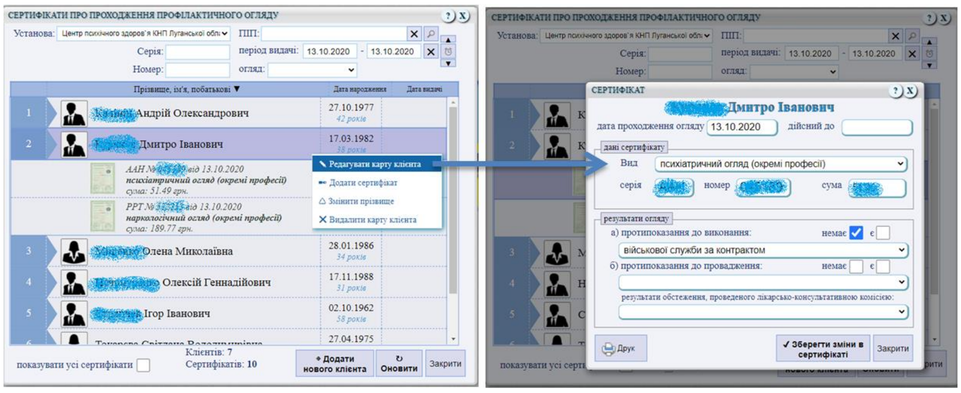
After the form is generated, the refreshBaza() method is called (Fig. 5). In it, an ajax request is sent to the server to the getSertifikat.php script, to the listFIO method of the class, which prepares the necessary data and organizes logic for the client side. The prepared information in the JSON object is delivered to the client browser, where it is converted to control parameters using an anonymous callback function. Form elements are accessed through the collObj collection of form objects.



```
Modul_Sertifikat.refreshBaza = function(){
let kodMed = this.formBaza.collObj['selectMed'].value;
    this.formBaza.WaitLoad(true);
    this.formBaza.wXHR.query({param:'listFIO', idustanova : kodMed,
                        FIO : formBaza.collObj['FIO'].value,
                        order : this.arOrderBaza.order, dir : this.arOrderBaza.dir}, 'php/getSertifikat.php');
    this.formBaza.wXHR.request.onreadystatechange = function(){
        if (this.readyState == 4 && this.status == 200){
            Modul_Sertifikat.formBaza.setHeight(700);
            Modul_Sertifikat.formBaza.WaitLoad(false);
            Modul_Sertifikat.formBaza.collObj['containerTabl'].style.visibility = 'visible';
            answer = this.responseText.split('|');
            Modul_Sertifikat.formBaza.collObj['tabl'].querySelector('tbody').innerHTML = answer[0];
            System.changeWidthTabl(formBaza.collObj['containerTabl'], formBaza.collObj['tabl'], 850, 831);
        }
    }
};
```

Fig. 5. Form data revision method

In Fig. 6 shows the result of generating a certificate base form with filled data. The JSON object received from the server was converted into form parameters: into a two-level table with a client base and a list of certificates belonging to individuals; to the assigned callback functions for the context menu events for selecting clients and working with certificates; to general information about the number of certificates for a given filter. When the client card editing item is selected, a new form with AJAX loading of client data from the server is generated.



СЕРТИФІКАТИ ПРО ПРОХОДЖЕННЯ ПРОФІЛАКТИЧНОГО ОГЛЯДУ

Установа: Центр психічного здоров'я КНП Луганської обл
ПІП:
Серія:
період видачі: 13.10.2020 - 13.10.2020
Номер:
огляд:

	Прізвище, ім'я, побатькові ▼	Дата народження	Дата видачі
1	[redacted] Андрій Олександрович	27.10.1977 42 років	
2	[redacted] Дмитро Іванович	17.03.1982 38 років	
	ААН № [redacted] від 13.10.2020 психіатричний огляд (окремі професії) сума: 51.49 грн.		
	РРТ № [redacted] від 13.10.2020 наркологічний огляд (окремі професії) сума: 189.77 грн.		
3	[redacted] Олена Миколаївна	28.01.1986 34 років	
4	[redacted] Олексій Геннадійович	17.11.1988 31 років	
5	[redacted] Ігор Іванович	02.10.1962 58 років	
6	Токарєва Світлана Володимирівна	27.04.1975	

показувати усі сертифікати ☐
Клієнтів: 7
Сертифікатів: **10**

СЕРТИФІКАТ

[redacted] Дмитро Іванович

дата проходження огляду 13.10.2020 дійсний до

дані сертифікату
Вид: психіатричний огляд (окремі професії)
серія [redacted] номер [redacted] сума [redacted]

результати огляду
а) протипоказання до виконання: немає ✓ є ☐
військової служби за контрактом
б) протипоказання до провадження: немає ☐ є ☐
результати обстеження, проведеного лікарсько-консультативною комісією:

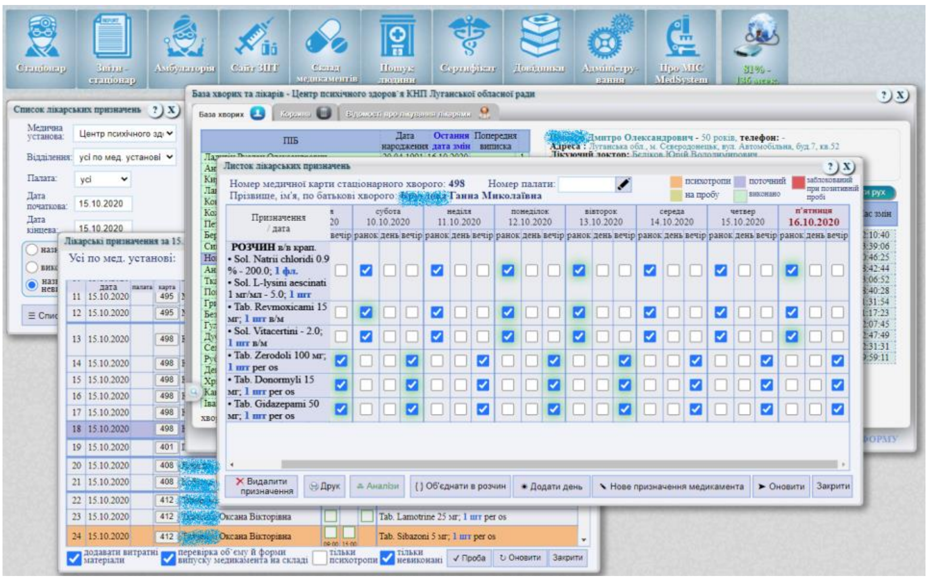
Fig. 6. Result of generating a certificate database form with filled data

To organize cloud computing and build an information system on the server side, the following software was used:

1. Server operating system – FreeBSD version 11.2.
2. Apache web server.
3. Hypertext preprocessor PHP version 7.4.
4. Database management system – MySQL server version 5.7.31.

To work on the client side of the software modules of an information system developed on the basis of JavaScript and Standard ECMA-262, any operating system is required that supports a www navigator with a JavaScript interpreter.

The practical implementation of the module model with the implementation of the dispatcher and the interactive window interface is represented by the installation in the form of the medical information system MedSystem being developed, in which the application modules are implemented in accordance with the considered interaction mechanism (Fig. 7).



База хворих та лікарів - Центр психічного здоров'я КНП Луганської обласної ради

Медична установа: Центр психічного здо
Відділення: усі по мед. установі
Палата: усі
Дата початкова: 15.10.2020
Дата кінцева: 15.10.2020

[redacted] Дмитро Олександрович - 50 років, телефон: -
Адреса : Луганська обл., м. Северодонецьк, вул. Автомобільна, буд.7, кв.52
Лікуючий доктор: Бєліков Юрій Володимирович

Листок лікарських призначень

Номер медичної карти стаціонарного хворого: 498 Номер палати:
Прізвище, ім'я, по батькові хворого: [redacted] Ганна Миколаївна

🟧 психотропи 🟪 поточний 🟥 заблокований при позитивній пробі
🟨 на пробу 🟩 виконано

Призначення / дата	20	субота 10.10.2020			неділя 11.10.2020			понеділок 12.10.2020			вівторок 13.10.2020			середа 14.10.2020			четвер 15.10.2020			п'ятниця 16.10.2020		
	вечір	ранок	день	вечір	ранок	день	вечір	ранок	день	вечір	ранок	день	вечір	ранок	день	вечір	ранок	день	вечір	ранок	день	вечір
РОЗЧИН в/в крап. • Sol. Natrii chloridi 0.9 % - 200.0; 1 фл. • Sol. L-lysini aescinati 1 мг/мл - 5.0; 1 шт	☐	✓	☐	☐	✓	☐	☐	✓	☐	☐	✓	☐	☐	✓	☐	☐	✓	☐	☐	✓	☐	☐
• Tab. Revmoxicami 15 мг; 1 шт в/м	☐	✓	☐	☐	✓	☐	☐	✓	☐	☐	✓	☐	☐	✓	☐	☐	✓	☐	☐	✓	☐	☐
• Sol. Vitacertini - 2.0; 1 шт в/м	☐	✓	☐	☐	✓	☐	☐	✓	☐	☐	✓	☐	☐	✓	☐	☐	✓	☐	☐	✓	☐	☐
• Tab. Zerodoli 100 мг; 1 шт per os	✓	☐	☐	✓	☐	☐	✓	☐	☐	✓	☐	☐	✓	☐	☐	✓	☐	☐	✓	☐	☐	✓
• Tab. Donormyli 15 мг; 1 шт per os	✓	☐	☐	✓	☐	☐	✓	☐	☐	✓	☐	☐	✓	☐	☐	✓	☐	☐	✓	☐	☐	✓
• Tab. Gidazepami 50 мг; 1 шт per os	✓	☐	☐	✓	☐	☐	✓	☐	☐	✓	☐	☐	✓	☐	☐	✓	☐	☐	✓	☐	☐	✓

Лікарські призначення за 15.10.2020
Усі по мед. установі:

	дата	палата	карта		
11	15.10.2020		495		
12	15.10.2020		495		
13	15.10.2020		498		
14	15.10.2020		498		
15	15.10.2020		498		
16	15.10.2020		498		
17	15.10.2020		498		
18	15.10.2020		498		
19	15.10.2020		401		
20	15.10.2020		408	[redacted]	
21	15.10.2020		408	[redacted]	
22	15.10.2020		412	[redacted]	
23	15.10.2020		412	[redacted] Оксана Вікторівна	Tab. Lamotrine 25 мг; 1 шт per os
24	15.10.2020		412	[redacted] Оксана Вікторівна	Tab. Sibazoni 5 мг; 1 шт per os

✓ додавати витратні матеріали ✓ перевірка об'єму й форми випуску медикамента на складі ☐ тільки психотропи ✓ тільки невиконані

Fig. 7. Medical information system MedSystem

4. CONCLUSIONS

The proposed module architecture template for the implementation of the dispatcher and the functionality of the user window interface made it possible to create the modules of the MedSystem medical information system. The modular approach had a positive effect on the responsiveness of the user interface and the ability to scale the functionality of the system itself when implementing cloud technologies.

The results of the use showed that the mechanism of modular creation of a dispatcher and a windowed interactive interface of web-forms not only organically fits into existing technologies for building web applications, but also itself has sufficient potential to become the core of cloud technologies for the development of multi-user information systems and web services.

REFERENCES

AngularJS – Superheroic JavaScript MVW Framework. (n.d.). Retrieved October 15, 2020 from http://angularjs.org

Backbone.js. (n.d.). Retrieved October 15, 2020 from http://backbonejs.org

Bootstrap. (n.d.). Retrieved October 15, 2020 from http://getbootstrap.com

Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification. (n.d.). Retrieved October 15, 2020 from https://www.w3.org/TR/CSS22/

Crane, D., & Pascarello, E. (2006). *Ajax in action.* Moscow: Ed. house "Williams".

Drupal – Open Source CMS. (n.d.). *drupal.org*. Retrieved October 15, 2020 from https://drupal.org

ECMAScript Language Specification – ECMA-262 Edition 5.1. (n.d.). Retrieved October 15, 2020 from http://www.ecma-international.org/ecma-262/5.1

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (2001). *Techniques for Object-Oriented Design. Design patterns.* SPb.: Peter.

HTML 4.01 Specification. (n.d.). Retrieved October 15, 2020 from https://www.w3.org/TR/html401

jQuery. (n.d.). Retrieved October 15, 2020 from http://jquery.com

jQuery UI. (n.d.). Retrieved October 15, 2020 from http://jqueryui.com

Knockout: Home. (n.d.). Retrieved October 15, 2020 from http://knockoutjs.com

MediaWiki. (n.d.). Retrieved October 15, 2020 from http://www.mediawiki.org/wiki/MediaWiki

Medvedev, A. (2013). Cloud technologies: development trends, examples of execution. *Modern automation technologies*, *2*, 6–9.

MVP architecture. (n.d.). Retrieved October 15, 2020 from http://www.gwtproject.org/articles/mvp-architecture.html

Papadopoulos, A., & Katsaros, D. (2011). Distributed Indexing of Multidimensional Data for Cloud Computing Environments. *Third IEEE Intl Conf. on Cloud Computing Technology and Science* (pp. 407–414). IEEE.

Stefanov, S. (2011). *Javascript. Patterns.* St. Petersburg: publishing house Symbol-Plus.

w2ui: Home (n.d.) JavaScript UI. Retrieved October 15, 2020 from http://w2ui.com/web

Zepto.js: the aerogel-weight jQuery-compatible JavaScript library. (n.d.). Retrieved October 15, 2020 from http://zeptojs.com

Applied Computer Science, vol. 16, no. 4, pp. 56–63
doi:10.23743/acs-2020-29

Submitted: 2020-11-23
Revised: 2020-11-30
Accepted: 2020-12-17

machine learning, EEG signal, classification, data balancing, feature extraction

Amina ALYAMANI [0000-0002-0286-7105]*,
Oleh YASNIY [0000-0002-9820-9093]**

CLASSIFICATION OF EEG SIGNAL BY METHODS OF MACHINE LEARNING

Abstract

Electroencephalogram (EEG) signal of two healthy subjects that was available from literature, was studied using the methods of machine learning, namely, decision trees (DT), multilayer perceptron (MLP), K-nearest neighbours (kNN), and support vector machines (SVM). Since the data were imbalanced, the appropriate balancing was performed by Kmeans clustering algorithm. The original and balanced data were classified by means of the mentioned above 4 methods. It was found, that SVM showed the best result for the both datasets in terms of accuracy. MLP and kNN produce the comparable results which are almost the same. DT accuracies are the lowest for the given dataset, with 83.82% for the original data and 61.48% for the balanced data.

1. INTRODUCTION

Hypnotic therapy is a method of psychotherapy that helps to heal a large number of disorders, including stress, depression, anxiety, pain (Provençal et al., 2018; Wood & Bioy, 2008), and eliminating the unwanted memories in patient mind (Terhune et al. 2017). Hypnosis can also enhance thought suppression by minimizing the effect of cognitive load (Bryant & Sindicich, 2007). The influence of hypnosis on the human being can be assessed by registering brain signals.

The widely used technique for recording brain signal is electroencephalogram (EEG) (Sanei & Chambers, 2007). EEG signal is a miniature amount of electrical flow in a human brain that holds and controls the entire body (Haykin 2009).

* Omar Al-Mukhtar University, Faculty of Engineering, Department of Computer Science, West Shiha, Dernah, Libya, amina.alyamani@yahoo.com
** Ternopil Ivan Pul'uj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Mathematical Methods in Engineering, Ruska 56, 46001, Ternopil, Ukraine, oleh.yasniy@gmail.com

EEG signal can be employed to diagnose Alzheimer disease (Podgorelec, 2012), to predict epileptic seizure (Satapathy, Jagadev & Dehuri, 2017), to detect mental disorders (Dvey-Aharon et al., 2015; Thilakvathi et al., 2017). The approaches of machine learning give the possibility to analyze the EEG signal and draw appropriate conclusions based on the results of performed analysis. In particular, various classification methods can help to diagnose the mentioned above diseases. In paper (Parvinnia et al., 2014), EEG signals were classified using adaptive weighted distance nearest neighbor algorithm. In the study (Amin et al., 2017), the pattern recognition approach was employed to classify the EEG signals. Often EEG signals contain artifacts which should be found and treated respectively. Paper (Lawhern et al., 2012) gives the methods to detect and classify the subject-generated artifacts in EEG signals using auto-regressive models. Results, obtained in the mentioned above study, indicate reliable classification among several different artifact conditions across subjects.

However, despite the numerous application of machine learning approaches to the EEG signals, there still remains a wide range of potential activity and many interesting problems to be solved by means of computer and respective algorithms.

The present study utilizes the dataset obtained in the frames of Horizon 2020 program, which is available at (Real & Kübler, 2014). EEG signals of two healthy subjects were recorded (S01: right-handed male; S02: right handed female) (Real & Kübler, 2014). The subject sat in a comfortable chair. Stimuli were presented in two conditions. In an active condition, the subject was told to listen to a tone stream, and that he/she should count the occurrence of the odd (low) tones. In a passive condition, the subject was told to listen to a series of tones and that he/she would just have to listen to the tones. After the first recording ("PRE"), the subject listened to an Erickson-type hypnotic induction, where the subject imagined being on a ship in dense fog, making hearing and seeing difficult. Then, the EEG experiment was repeated (datasets "POST"). Finally, the subject listened to instructions designed to take back a hypnotic state (Real & Kübler, 2014).

2. METHODOLOGY

2.1. Dataset preparation

The dataset S01 No5 consists of four EEG data matrices, *X*s, where the columns of each *X* show outputs of 31 recording electrodes and rows of each *X* contain recorded data in every 0.001953125 second (or 0.002 for simplification) from each electrode. Since the sampling rate is 512 Hz, the sampling period is around 0.002 second. Also, the duration of each stimuli is 50 ms, which means that every data-point is a 25 by 31 submatrix (each stimulus is 50 ms and by dividing it to the sample period 2ms on gets the number 25).

The dataset provides the time period when each stimulus is presented. This time period is available in the vector ‘trial’. The class label of each stimuli is available in the vector ‘y’. So, it is possible to find each data-point along with its label. For example, the first stimuli starts at time period 19898 ms, so to find the EEG response of this stimuli one needs to extract the submatrix of *X*, i.e. the row number 19898÷2 of matrix *X* to row number (19898÷2 + 25) of matrix *X*.

Also, the label of this data-point is provided in the first entry of vector ‘y’ in data cell number 1 of the S01 file. This label is ‘2’ which means that this stimulus is from class ‘frequent’. There are two classes of data: ‘odd’ and ‘frequent’. The ‘odd’ class is represented with label ‘1’.

There was written the code to perform the mentioned-above procedure in order to prepare the dataset. This code reads all datasets S01 and S02 and concatenates them according to the described procedure.

After the data preparation step was performed and for the better understanding of the samples of dataset, 9 samples of the dataset were visualized from both male and female data. Fig. 1 shows the visualized results. In each subplot, it is possible to see one sample of stimuli along with its label on the top. Each sample consists of 31 stimuli which have been drawn with different colors. The horizontal and vertical axes show time period and amplitude of the stimuli; respectively. As it is possible to see, the amplitude value of each stimulus changes with time and this change is significant. Another important point is that it can be observed from the figure that there are some distinct patterns in the behaviors of stimuli. For the male data, i.e. S01.mat, the stimuli of the ‘odd’ class somehow shows a descending behavior during the time while the stimuli of the other class show an ascending and then descending behavior in their shapes. Similarly, it is possible to note some distinctive patterns in the female data, i.e. S02.mat.

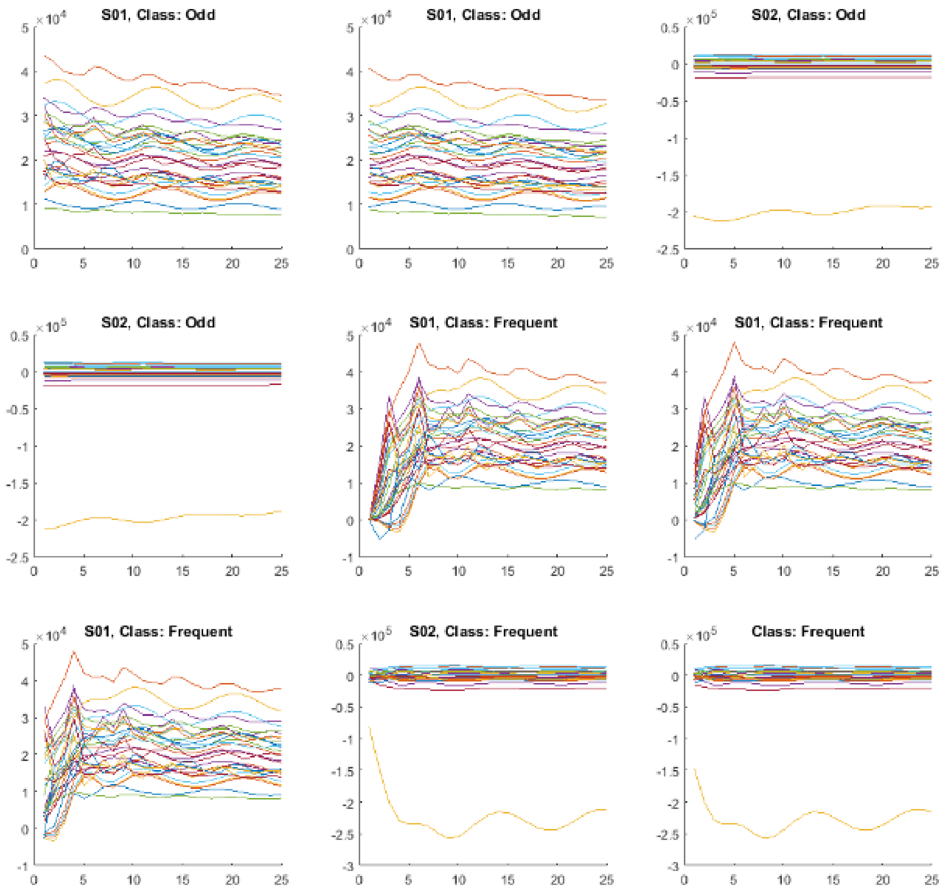


Fig. 1. A graphical view of the stimuli samples of dataset no. 5 along with their labels

2.2. Feature extraction

Features extraction is a common way to extract meaningful features from the EEG data (Li et al., 2015; Sun et al., 2019). In this way, an Autoencoder was used to find a new representation of the data in a lower dimensional space. Autoencoder is an unsupervised method of machine learning that provides a new representation of data in a lower dimensional space (Hinton & Salakhutdinov, 2006). In other words, an autoencoder is a type of artificial neural network used to learn efficient data encoding in an unsupervised manner. The aim of the Autoencoder is to learn a representation for a set of data, typically for dimensionality reduction. Along with the reduction side, a reconstructing side is learnt, where the autoencoder tries to generate the representation as close as possible to its original input from the reduced encoding.

The following model is used for the autoencoder network. The input size was $25 \cdot 31$, which is equal to 775. Hidden layer size is 64. Output layer size was $25 \cdot 31$, that is the same as input size.

2.3. Balancing the data

Another difficulty of the data is that it is an imbalanced data. In other words, the ratio of its classes is highly different that comes from the fact that the 'frequent' class has much more data than the 'odd' class. Originally, there are more than 3000 data points for the 'frequent' class and 480 data points for the 'odd' class of data, i.e. 3000 >> 480.

In this experiment, two scenarios were employed. In the first scenario, the original data were used. In the second scenario, there was an attempt to turn data into a balanced data. Both scenarios were implemented and the results were presented:

1. For the first scenario the data were not changed and all of the samples were kept.
2. In the second scenario, an under-sampling technique was employed to turn the data into a balanced one. To accomplish that, Kmeans clustering algorithm (MacQueen, 1967) was applied over the data points of the 'frequent' class. The number of clusters was set to 1000, and Kmeans algorithm was started. After the learning, the centers of clusters were treated as the new data points for the 'frequent' class of data. The number 1000 was determined empirically.

After that step, 1000 data points were obtained for the 'frequent' class of data and 480 data points were for the 'odd' class of data. Although the dataset is not exactly balanced, nevertheless, this ratio of classes samples is more balanced than the original one.

2.4. MLP structure

The multilayer perceptron (MLP) neural network was employed which uses the gradient decent back-propagation for tuning the parameters of the network (Haykin, 2009). The MLP topology is the following.

Input size, that is the number of nodes in input layer was 64, number of hidden layers was 1, number of neurons in each hidden layer was 32, number of neurons in output layer was 1.

3. RESULTS

To verify the models, 10-fold cross validation method was performed. In this way, the data were split into ten parts and each time one part was taken as the test data and the rest was treated as training data.

The classification was performed by four methods: support vector machines (SVM) (Cortes & Vapnik, 1995), Decision Trees (Quinlan, 1986), K-nearest neighbors (Altman, 1992) and MLP (Haykin, 2009). Table 1 presents the results in terms of accuracies along with the standard deviations. According to the results shown in Table 1, it can be seen that SVM performs better in both datasets and provide higher accuracies with accuracy 87.47% for the original data and 66.95% for the balanced data. Also, MLP and kNN produce the comparable results which are almost the same. DT accuracies are the lowest for the given dataset, with 83.82% for the original data and 61.48% for the balanced data.

Tab. 1. Comparison of different methods in terms of accuracies

Dataset	SVM	kNN (k = 8)	DT	MLP
Original data	87.47 ± 1.51	87.108 ± 1.56	83.82 ± 1.58	87.03 ± 1.77
Balanced data	66.95 ± 1.89	64.05 ± 4.03	61.48 ± 3.24	65.74 ± 2.06

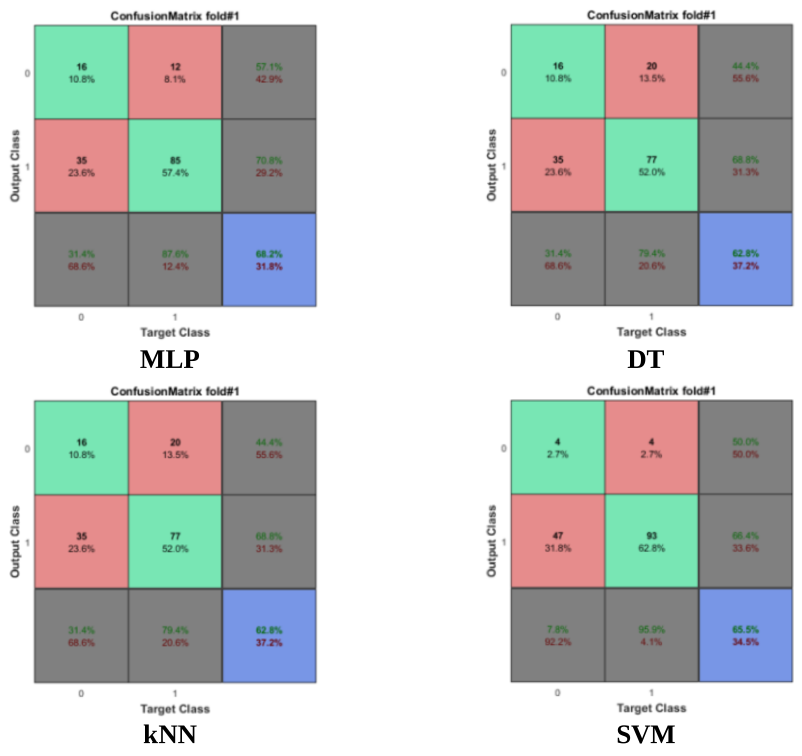


Fig. 2. Confusion matrices over the balanced data (Fold #1)

Fig. 3 presents the confusion matrices over the original dataset, i.e. the imbalanced version of dataset.

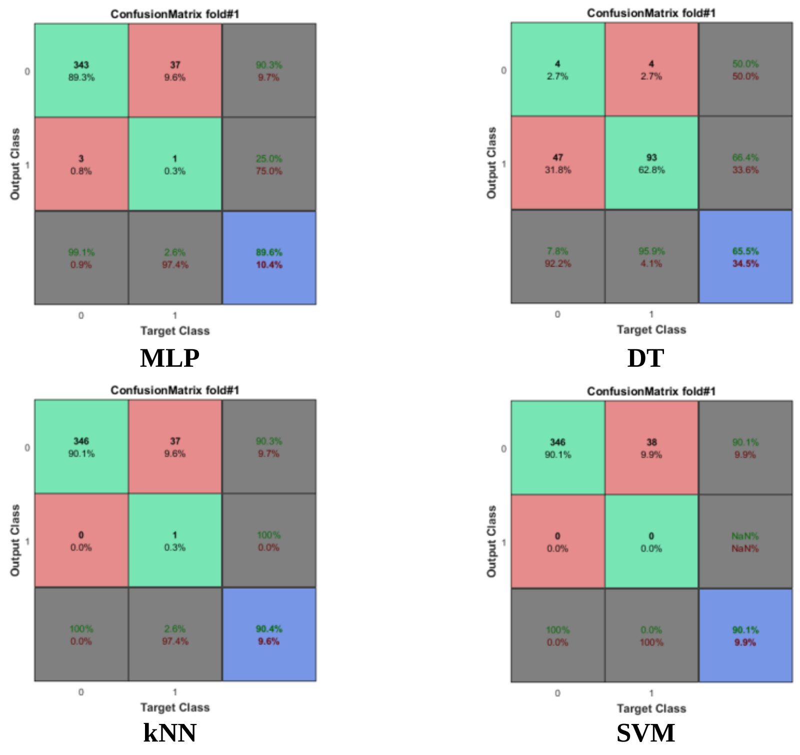


Fig. 3. Confusion matrices results over the original data (Fold #1)

The confusion matrices obtained over the original data have better values due to the imbalanced data. However, the balancing makes the accuracy of classification less as compared with the original data, though allows to change the ratio of classes samples, so the data are more balanced than the original one.

4. CONCLUSIONS

EEG signal of two healthy subjects was studied using the methods of machine learning, namely, decision trees (DT), multilayer perceptron (MLP), K-nearest neighbours (kNN), and support vector machines (SVM). Since the data were imbalanced, the appropriate balancing was performed by Kmeans clustering algorithm. The original and balanced dataset was classified by means of the mentioned above 4 methods. The obtained classification results were compared. It was found, that SVM showed the best result for the both datasets in terms of accuracy. This method gave 87.47% accuracy for the original data and 66.95% accuracy for the balanced data. MLP and kNN produce the comparable results which are almost the same. DT accuracies are the lowest for the given dataset, with 83.82% for the original data and 61.48% for the balanced data. The respective confusion matrices were also built.

In general, the methods of machine learning allow classifying the EEG signals and obtaining rather accurate results.

REFERENCES

Altman, N. S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician, 46*(3), 175–185.

Amin, H. U., Mumtaz, W., Subhani, A. R., Saad, M. N. M., & Malik, A. S. (2017). Classification of EEG Signals Based on Pattern Recognition Approach. *Frontiers in Computational Neuroscience*, *11*(103), 1–12.

Bryant, R. A., & Sindicich, N. (2007). Hypnosis and Thought Suppression – More Data: A Brief Communication. *International Journal of Clinical and Experimental Hypnosis*, *56*(1), 37–46.

Cortes, C., & Vapnik, V. N. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297.

Dvey-Aharon, Z., Fogelson, N., Peled, A, & Intrator, N. (2015). Schizophrenia Detection and Classification by Advanced Analysis of EEG Recordings Using a Single Electrode Approach. *PLoS ONE, 10*(4), 1–12.

Haykin, S. (Ed.). (2009). *Neural Networks and Learning Machines (3rd Edition).* New Jersey, Prentice Hall.

Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, *313*(5786), 504–507.

Lawhern, V., Hairston, W. D., McDowell, K., Westerfield, M., & Robbins, K. (2012). Detection and classification of subject-generated artifacts in EEG signals using autoregressive models. *Journal of Neuroscience Methods*, *208*(2), 181–189.

Li, J., Struzik, Z., Zhang, L., & Cichocki, A. (2015). Feature learning from incomplete EEG with denoising autoencoder. *Neurocomputing*, *165*, 23–31.

MacQueen, J. B. (1967). Some Methods for classification and Analysis of Multivariate Observations. *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability – Volume 1: Statistics*, 281–297.

Parvinnia, E., Sabeti, M., Zolghadri Jahromi, M., & Boostani, R. (2014). Classification of EEG Signals using Adaptive Weighted Distance Nearest Neighbor Algorithm. *Journal of King Saud University – Computer and Information Sciences, 26*(1), 1–6.

Podgorelec, V. (2012). Analyzing EEG signals with machine learning for diagnosing Alzheimer’s disease. *Elektronika i Elektrotechnika*, *18*(8), 61–64.

Provençal, S. C., Bond, S., Rizkallah, E., & El-Baalbaki, G. (2018). Hypnosis for burn wound care pain and anxiety: A systematic review and meta-analysis. *Burns*, *44*(8), 1870–1881.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, *1*, 81–106.

Real, R. G. L., & Kübler, A. (2014). Auditory oddball paradigm during hypnosis. *Institute of Psychology, University of Würzburg*.

Sanei, S., & Chambers, J. A. (Eds.). (2007). *EEG Signal processing*. Great Britain, Chippenham, John Wiley & Sons.

Satapathy, S. K., Jagadev, A. K., & Dehuri, S. (2017). Weighted majority voting based ensemble of classifiers using different machine learning techniques for classification of EEG signal to detect epileptic seizure. *Informatica*, *41*(1), 99–110.

Sun, L., Jin, B., Yang, B., Tong, J., Liu, C., & Xiong, H. (2019). Unsupervised EEG Feature Extraction Based on Echo State Network. *Information Sciences*, 475, 1–17.

Terhune, D. B., Cleeremans, A., Raz, A., & Lynn, S. J. (2017). Hypnosis and top-down regulation of consciousness. *Neuroscience and Biobehavioral Reviews*, *81*(A), 59–74.

Thilakvathi, B., Shenbaga, Devi, S., Bhanu, K., & Malaippan, M. (2017). EEG signal complexity analysis for schizophrenia during rest and mental activity. *Biomedical Research*, *28*(1): 1–9.

Wood, C., & Bioy, A. (2008). Hypnosis and Pain in Children. *Journal of Pain and Symptom Management*, *35*(4), 437–446.

Applied Computer Science, vol. 16, no. 4, pp. 64–84
doi:10.23743/acs-2020-30

Submitted: 2020-09-16
Revised: 2020-09-22
Accepted: 2020-10-05

Felder-Silverman Learning Style Model, Item response theory, Ontology, Learning ability, Difficulty level

Olutayo BOYINBODE [0000-0002-6789-258X]*, *Paul OLOTU**, *Kolawole AKINTOLA**

DEVELOPMENT OF AN ONTOLOGY-BASED ADAPTIVE PERSONALIZED E-LEARNING SYSTEM

Abstract

E-learning has fast become an active field of research with a lot of investments towards web-based delivery of personalized learning contents to learners. Some issues of e-learning arise from the heterogeneity and interoperability of learning content adapting to learner's styles and preferences. This has brought about the development of an ontology-based personalized learning system to solve this problem. This research developed an ontology-based personalized e-learning system that presents suitable learning contents to learners based on their learning style, preferences, background knowledge, and personal profile.

1. INTRODUCTION

Learning is enormously affected by the improvement of Information and Communication Technologies and informed computerized media. E-learning enables access to the training of individuals who think that it's hard to be physically present in the customary study of hall-based learning (Boyinbode & Akintade, 2015; Uhomoibhi, 2006). Personalization is said to exist where training programs are customized to individual learners, based on an analysis of the learners' objectives, current status of

* The Federal University of Technology, School of Computing, Department of Information Technology, FUTA Rd, Akure, Nigeria, okboyinbode@futa.edu.ng, pkootu@futa.edu.ng, kgakintola@futa.edu.ng

skills/knowledge, learning style preferences, as well as constant monitoring of progress. Online learning material can be compiled to meet personal needs, capitalizing on re-usable learning objects (Boyinbode & Bagula, 2012).

Some educational issues are taken care of normally through the presentation of a personalized adaptive e-learning system (Adewale, 2006). This system encourages students to learn effectively based on their style of learning and enhance improvement in the performance of the learners. The adaptability of the E-learning platform encourages students to learn with their most preferred method of learning and finish their courses effectively (Adewale, 2006).

2. REVIEW OF RELATED WORK

Kurilovas et al. (2016) developed a personalized learning system based on students' learning styles and application of intelligent technologies, where learners have different features and characteristics such as prior knowledge, intellectual level, interests, goals, cognitive traits (working memory capacity, inductive reasoning ability, and associative learning skills), there came a need for learning behavioral type (according to his/her self-regulation level) and finally learning styles.

The system was designed to perform a systematic review of learning personalization; identify a student with certain learning style, according to felder and silver man learning style model (FSLSM) and finally create a model of personalized intelligent learning system based on students' learning styles, cognitive traits, and other personal characteristics and needs. FSLSM is recognized to be the most suitable for STEM (Science, Technology, Engineering, and Mathematics) and e-learning. Dedicated psychological questionnaire – Sloman and Felder's Index of Learning Styles is used to explore students' learning styles according to FSLSM. The research does not include the creation of pedagogically sound vocabularies of the learning components.

Funda and Aynur (2009) analyzed relations between online learning and learning styles; researchers have investigated that presentation of learning content and learning tools are based on learning styles in the online learning, environments are a factor which impacts the academic achievements of the learner. In the other research approach, researchers have used learning styles as a supportive factor to design the online learning environments for personalized online learning. The hybrid of these research approaches was adopted which suggested that improving the academic achievements of the learners can be achieved by considering the motivation of the learner, demographics factors, teaching strategies, and teaching methods.

Latha and Kirubakaran (2013) presented a Personalized Learning Path Delivery in Web-based Educational Systems using a Graph Theory-based Approach. The absence of a teacher or trainer becomes a bottleneck inappropriately delivering contents to the learner. Developing a system with a novel way of recommending a personalized learning path to a user became important; a graph theory-based approach in web-based learning systems was adopted to make the learning process effective.

Agbonifo and Obolo (2018) developed a Genetic Algorithm-based Curriculum Sequencing Model for Personalised E-Learning System, in which the difficulty level and the relationship degree that exists between various course concepts were recorded to affect the learning ability and the overall performance of the learner. The research focused on enabling the learner to identify the difficulty level of each course concept or curriculum and the relationship degree that exist between them to provide optimal personalized learning pattern to the learner to improve their performance.

Yarandi et al., (2013) proposed an adaptive e-learning approach based on semantic web technology; it is becoming increasingly difficult to ignore adaptation in the field of e-learning systems. Many researchers are adopting semantic web technologies to find new ways for designing adaptive learning systems based on describing knowledge using ontological models; this motivated the development of a personalized adaptive e-learning approach based on semantic (ontology) web technology.

3. METHODOLOGY

Ontology is characterized as a representation of a phenomenon's dynamic model on the world using conceptualization, which helps with distinguishing the allotment of area ideas, using formal definitions regarding adages and the ideas' semantic connections (Chi, 2009). Information portrayal utilizing ontologies encourages sorting out the metadata of complex data assets.

These metadata give syntactic and semantic data about data assets which are encoded as examples in the cosmology. Differential Equations are characterized as ideas or classes. W3C Web Ontology Language (OWL) is a Semantic Web language designed to represent rich and complex knowledge about things, groups of things, and relations between things. The OWL file obtained from the protégé tool is used to extract the concepts or classes that are represented in a specific domain through the domain ontology. These concepts are saved in a vector denoted as $C= [c1, c2, c3..., cm]$ to determine similarities with the XHTML files produced from HTML files. The algorithm used for the extraction of OWL concepts is given:

Algorithm:

```
Ontology concept extraction
Input: OWL Ontology Document
Output: Vector of Ontology Concepts (C)
BEGIN
1.  Declare Vector (C), OWL Ontology Document, Xpath;
2.  Define XPATH to get the Ontology concepts from the input OWL Ontology
Document
3.  Pass ontology concepts and store into (C)
4.  Return Vector of Ontology Concepts (C)
END
```

3.1. System Architecture

The ontology-based adaptive personalized e-learning system proposed consists of the following major components as shown in Figure 1.

3.1.1. User Interface

This gives a versatile and easy to use interface for communication with learners. The interface connects user features to the user model ontology, and enables sending the adaptive content from the Adaptive Engine to the user. The user interface additionally sends back the user's reactions to the adaptive engine. For a start-up user, there is an enrollment cycle, where the general and instructive attributes of the user are taken and recorded into the ontological based user model.

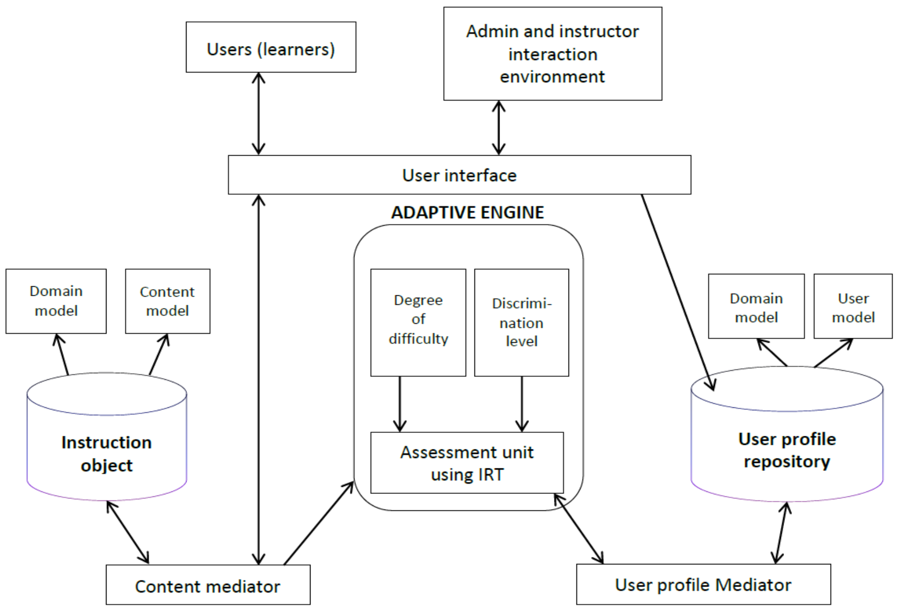


Fig. 1. The Architecture of the System

3.1.2. Personalized Adaptive Engine

This signifies the powerhouse of the e-learning structure which is responsible for presenting personalized learning content anchoring on the material available in the learner's model. The engine merges up instruction objects to produce particular and structured learning content for a particular learner. It obtains facts about learners and learning objects with associated mediators. The engine is also an evaluation element to re-evaluate the stage of knowledge and ability of learners.

This section will subject learners to regular tests and evaluates their performance in the selected topic and also learner's ability based on the item response theory. The user model is updated on the note of the evaluated information acquired from the result of the assessments, which will redefine the profile of the user.

3.1.3. User Profile Mediator

The Mediator is liable for the management of any form of requests, for opening and modernizing the user model repository.

3.1.4. Content Mediator

The Content Mediator is in control of examining the repository and retrieving diverse kinds of instruction objects depending on the diverse instructional role. This mediator also conforms the retrieved Instruction objects into Lessons and marks lessons spontaneously.

The construction comprises two repositories namely Instruction Objects and user profiles. The Instruction Object repository comprises of all learning contents and their metadata based on the content model ontology.

3.1.5. User Profile Repository

This is where the user profile and activities are stored. It houses all users' actions on his/her interfaces.

3.1.6. Domain Model

The domain model is a semantic ontology which is determined by the course creator and structures a coherent scientific classification for the information area. It indicates the subject order of learning objects. The domain ontology contains classes and properties, that portray subjects of an area and educational relationship, between proposed titles or topics. In this system, General Studies Course (GNS) is used for the system.

3.1.7. User Model

The system designs an ontological user model that design the user profile. It includes all the properties of the user(learner). The learner's properties are arranged in two groups including user identification information and learning profiles. User identification information such as names, date of birth, sex, passwords, and emails are kept in the personal information class through data properties which are attached to this class. Other classes and properties of this ontology are designed to characterize the learner's learning profiles such as preferences, learning performance, learning abilities, and learning styles.

The individual learner will also be attached to a set of performance-related data that is presented in performance class via has performance property. Learning performance which contains prior knowledge and gained knowledge can be obtained as a result of technical examination which is taken by individual learners. Ability class will represent the abilities of learners, which are calculated according to item response theory during the learning process. The learning styles of individual learners are recorded in the learning style class based on the Felder-Silverman Learning Style Model (Brusilovsky et al., 2005). This model defines four dimensions namely active-reflective, visual-verbal, sensing-intuitive, and sequential-global for a particular learner. The learning style class presents these dimensions through the learning category class. The learning style of each learner is determined through the result of a questionnaire based on the Felder and Silverman's learning style model.

The learning ability of the learner is calculated using item response theory according to Chen and Chung (2008), which also confirms, that the difficulty level of the recommended content is extremely relevant to learners' abilities. Additionally, the wrong content can result in learner's intellectual confusion in learning practice. In the first step, the learner's ability initiates at a moderate level. In different levels of learning, tests are taken from individual learners regularly and their response is analyzed according to the Item Response Theory (Baker, 2001), which will dynamically estimate and update learners' abilities. In the next level, the right content is recommended based on the updated abilities.

Item response theory is a model-based method designed to choose the most suitable items for learners based on accurate relationships between abilities and item responses. Item response theory is built on the postulation that the likelihood of a correct response to an item is a mathematical function of personalized and itemized variables. The element variable is considered as the item difficulty, item discrimination, and the effect of random guessing. (Baker, 2001).

$$P_i(\phi) = c_i + (1 + c_i)\frac{1}{1+\exp(-a_i(\phi - b_i))}. \tag{1}$$

$P_i(\phi)$ is the probability that an examinee with ability ϕ can respond correctly to the item i. The three-parameter logic function is adapted where:
b_i is the difficulty parameter of item i,
a_i is the discrimination degree of item i,
c_i is the guessing degree of item i,
ϕ is the ability level of the learner.

In this methodology, the item parameters are kept in the Item class of content ontology through some data properties such as the difficulty, discriminations, guessing, etc.

To evaluation, the ability of a learner, the answers of the learner for all items of an exam are distinctly scored. This means that the learner has 1 for a unique answer scored correctly and 0 for the answer gotten wrongly. Hence, there is a response pattern of the form (U_1, U_2, U_3 ... U_j ... U_n) known as test response vector, where $U_j = 1$

is known for a correct answer gotten by the learner for the j^{th} item in the exam. On the contrary, $U_j = 0$ signifies a wrong answer gotten by the learner for the j^{th} item in the exam (Hambleton, Swaminathan & Rogers, 1991). Bock derived the quadrature form to estimate the learner's ability (Baker, 1992):

$$\phi = \frac{\sum_k^q \phi L(u_1, u_2, \dots, u_n|\phi) A(\phi_k)}{\sum_k^q L(u_1, u_2, \dots, u_n|\phi) A(\phi_k)}, \tag{2}$$

where ϕ is the estimation of the ability of the learner, $L(u_1, u_2, \dots, u_n|\phi)$ is the value of likelihood function and $A(\phi)$ represents the quadrature weight at a level below the learner's ability.

$$L(\phi|u_1, u_2, \dots, u_n) = \prod_{i=1}^n P(\phi)^{u_1} \, Q(\phi)^{(1-u)}, \tag{3}$$

where $P_i(\phi)$ represents the chances that the learner answers correctly to the i^{th} item at a level below his ability level ϕ, $Q_i(\phi) = 1 - P_i(\phi)$ signifies the likelihood that the learner answered inaccurately to the i_{th} item at a level below the ability level, $, u_i = 1$ if the result of the i_{th} item is correct and $, u_i = 0$ if the response of i_{th} item is inappropriate (Chen & Chung, 2008). To calculate the difficulty level of the course items, Crocker, and Algina (1986) was adapted:

$$P_i = \frac{A_i}{N_i}, \tag{4}$$

where P_i is the difficulty index of item i, A_i is the number of the correct answer to item i, and N_i is the number of correct answers plus the number of the incorrect answers to item i.

The difficulty of an item is understood as the proportion of persons who answer a test item correctly, the higher this proportion the lower the difficulty level and vice versa. The discrimination level of the items;

$$a_i = D_i = \frac{GH\ correct\ answer - GL\ correct\ answer}{0.5 * N_{large\ group}}, \tag{5}$$

where D_i the discrimination index of item i. $GH\ correct\ answer$ is the number of the correct answer to item i among those with the highest test score. $GL\ correct\ answer$ is the number of the correct answer to items i among those with the lowest test score.

The discrimination level of an item is normal if it’s approaching one, hence the item is acceptable, else if the discrimination level of an item is approaching zero, the item is poor and unacceptable.

The guessing degree is calculated by adding up the number of points earned by all learners on an item and divides it by the total number of learner (Abu-Sayf, 1979):

$$G = \frac{P_{total}}{L_{total}}, \tag{6}$$

where G is the guessing degree, P_{total} is the total number of points by the learner and L_{total} is the total number of the learner.

Guessing is discouraged utilizing instructions given on the test and by scoring the test in such a way as to penalize those who guess incorrectly by the use of formula scoring (correction for guessing). Though the procedure has been a source of controversy for many years (Hamzeh, 2005):

$$S = R = \frac{W}{A-1}, \tag{7}$$

where S represents the corrected score, R represents the number of right answers, W represents the number of wrong answers, A represents the number of alternatives per item.

Item Response Theory is used in the high-tech adaptive test to define the best items for learners based on their distinct abilities. Currently, the Computerized Adaptive Testing (CAT) concept has been successfully used in many real applications such as GMAT, GRE, and for the TOEFL.

4. IMPLEMENTATION AND RESULTS

This section defines the implementation of a personalized adaptive e-learning system. The system interface is displayed upon the successful launch of the page.

4.1. Registration Page

The registration page allows new learners to register using the registration form before login in, it's done using the “sign up menu” (Figure 3).

4.2. Home Page

Displays the first page the user comes in contact with when he/she successfully logs into the system, it is the page, where the user signs up and logs in. This is to ensure that only registered and valid users are allowed to perform certain tasks in the portal. A learner can also register and login to the portal to check the delivered content suitable for them based on their learning style (Figure 4).

4.3. FSLSM Learning Style Detector Page

This contains a catalog of questions, each first-time user answers to detect their learning style to enhance the right delivery of content (Figure 5).

4.4. The Dashboard Page

This page contains an overview of the list of departments, and courses present in the system.

4.5. Examination and Test Score Page

This page is the interface for examination concerning the course taking and also helps in displaying the examination scores of the user (Figure 6).

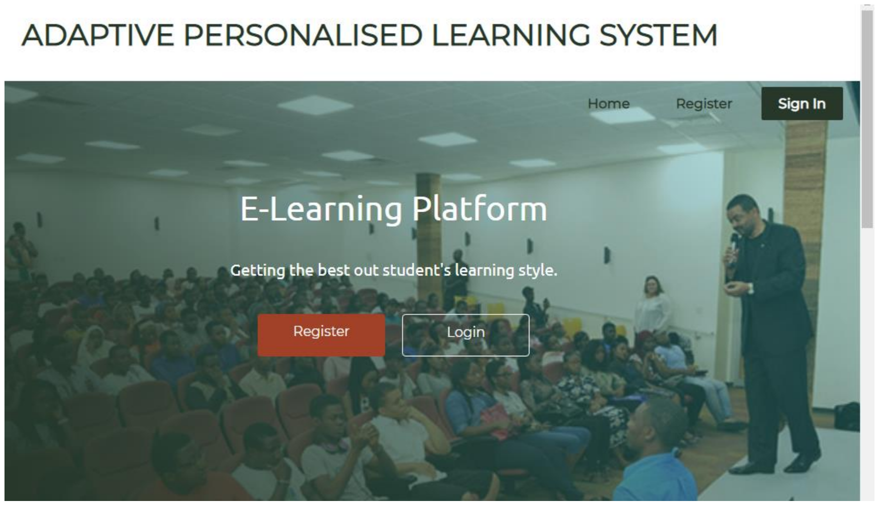
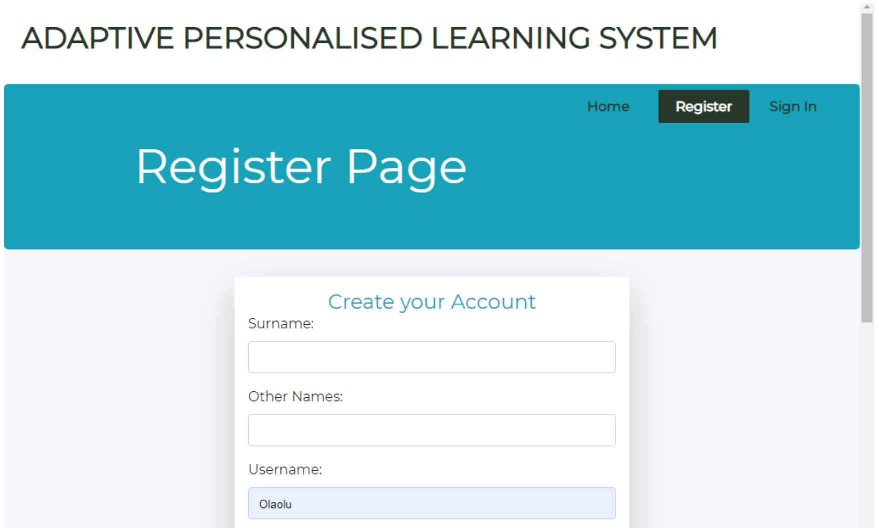

ADAPTIVE PERSONALISED LEARNING SYSTEM
Home
Register
Sign In
E-Learning Platform
Getting the best out student's learning style.
Register
Login

Fig. 2. System Home Page

Users access the system by registering into the system in other to generate the username and password for the user to login with into the system.


ADAPTIVE PERSONALISED LEARNING SYSTEM

Register Page

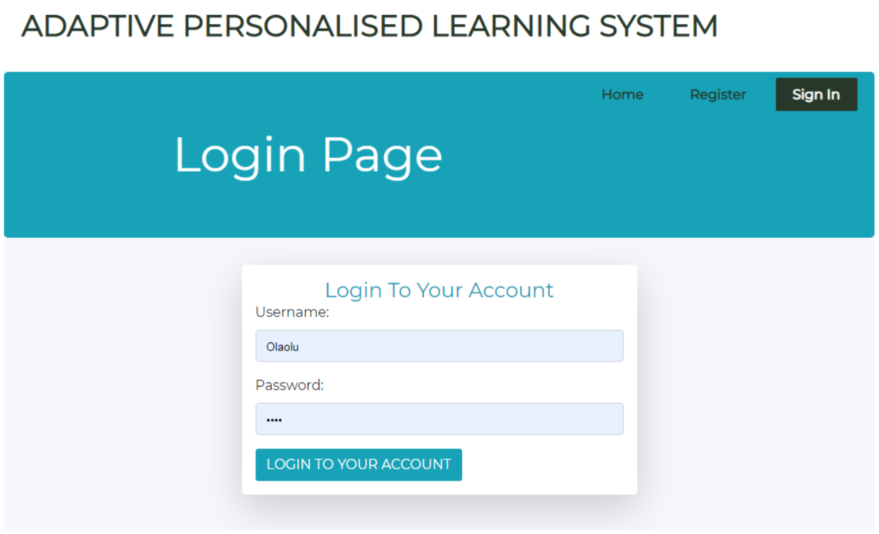
Create your Account

Surname:

Other Names:

Username: Olaolu

Fig. 3. Register page


ADAPTIVE PERSONALISED LEARNING SYSTEM

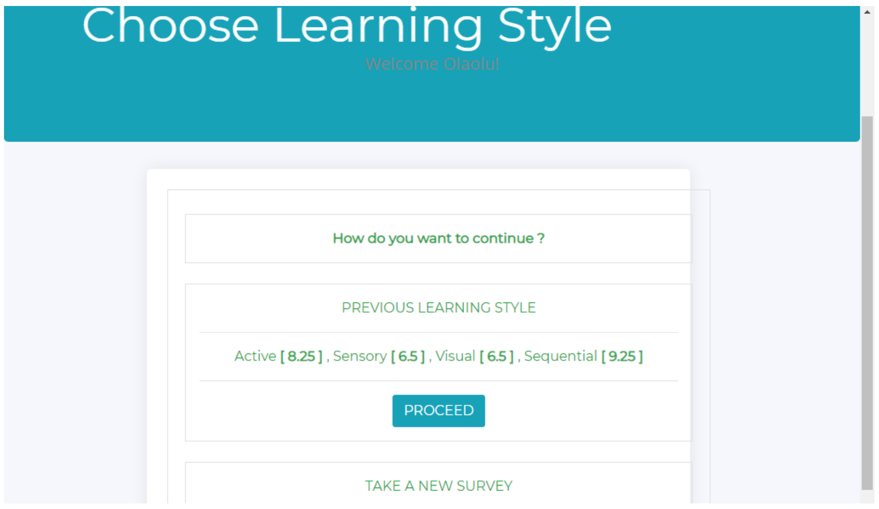
Login Page

Login To Your Account

Username: Olaolu

Password: ····

Fig. 4. Showing the Login page for the system


Choose Learning Style

Welcome Olaolu!

How do you want to continue ?

PREVIOUS LEARNING STYLE

Active [8.25] , Sensory [6.5] , Visual [6.5] , Sequential [9.25]

Fig. 5. Showing the record of the learning styles

For a new user, it is mandatory to run a survey which will help the system to capture the learning style of such user, and for an existing user, the learning style has been captured and saved and the user can easily continue with the saved learning style and also has the option of retaking the survey to confirm his or her learning style (Figure 5).In the dashboard of the system, different functions are displayed and the user can easily navigate through the system to enroll (Figure 6).

The user browses courses to check the courses available for enrollment. At the enrolling stage, the user is expected to enroll for a 'beginner' as the proficiency level, because the courses are designed ontologically such that the beginner has courses arranged for that category base on the difficulty level of those courses, which after successful completion of the beginner level, an examination that will show the eligibility of the user to move to the next level, which is the intermediate level is delivered to the user (Figure 7 and 8). Also, there is provision for new users that claim to be at the intermediate or expert level to take an examination of the previous level to determine his/her fitness for that level. A brief examination summarizing the knowledge of the beginner level is given for the intermediate level.

A concise examination for intermediate level is delivered, for the expert level. The eligibility of the user for the level will be determined; if the user failed the examination, he/she cannot proceed to the next level. The system will communicate to the user that he/she is not qualified for the level claimed, please go for the beginner level (Figure 9).

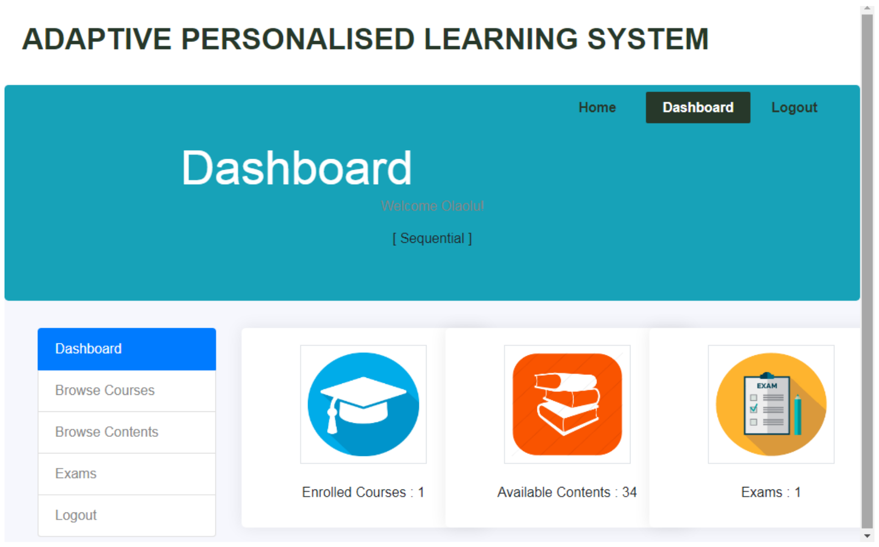


Fig. 6. Showing the Dashboard for the system

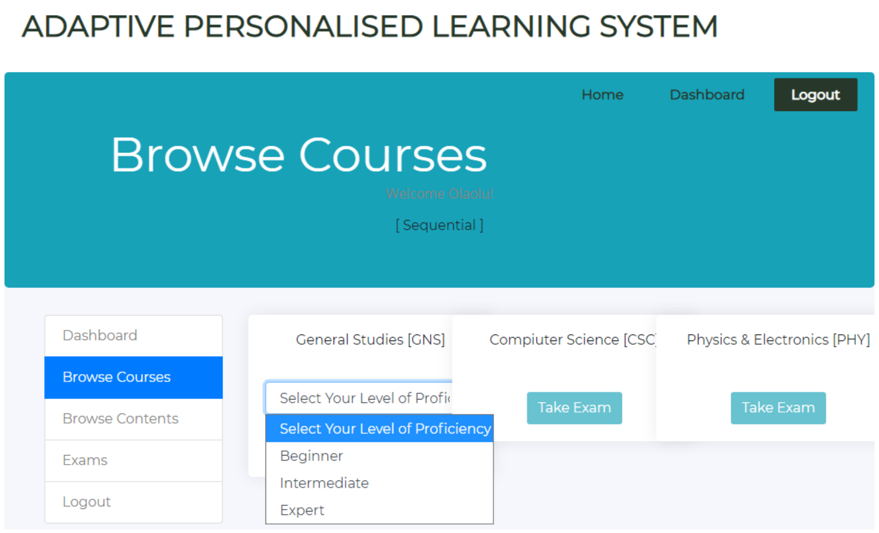
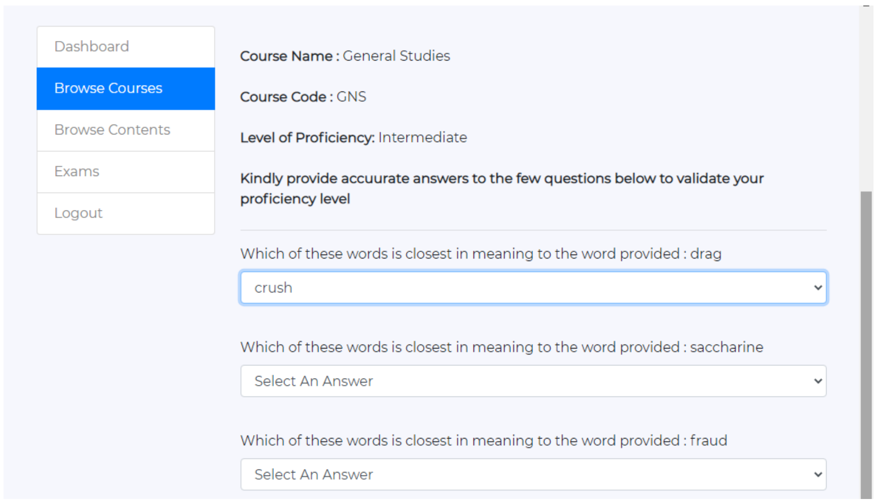


Fig. 7. Showing the Learning Categories



Course Name : General Studies

Course Code : GNS

Level of Proficiency: Intermediate

Kindly provide accuurate answers to the few questions below to validate your proficiency level

Which of these words is closest in meaning to the word provided : drag

crush

Which of these words is closest in meaning to the word provided : saccharine

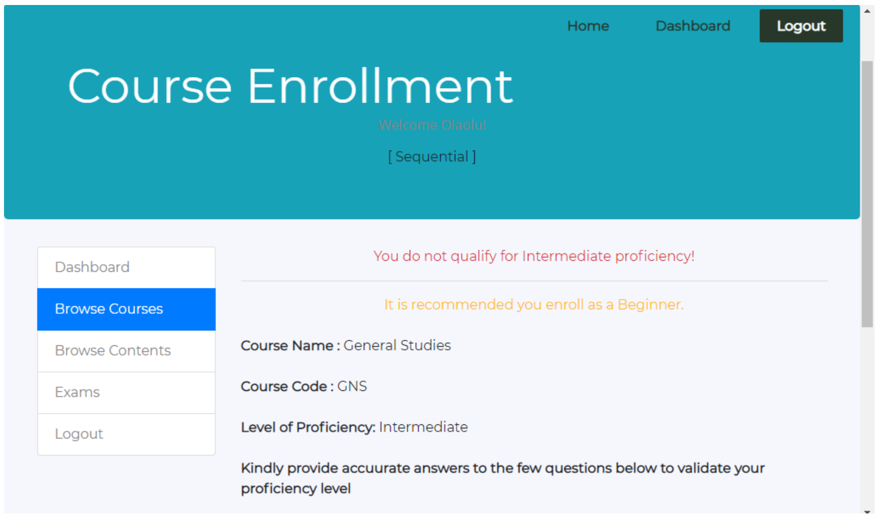
Select An Answer

Which of these words is closest in meaning to the word provided : fraud

Select An Answer

Fig.8. Showing the Eligibility Test Page

Also, if the user is not eligible for the proficiency level he or she claims, it will be revealed in the performance of the user in the eligibility test. The user has just two times, to attempt the eligibility test after which if the user failed, the system will recommend the user to start from the beginner level of the course (Figure 9).



Course Enrollment

Welcome Olaolu!

[Sequential]

You do not qualify for Intermediate proficiency!

It is recommended you enroll as a Beginner.

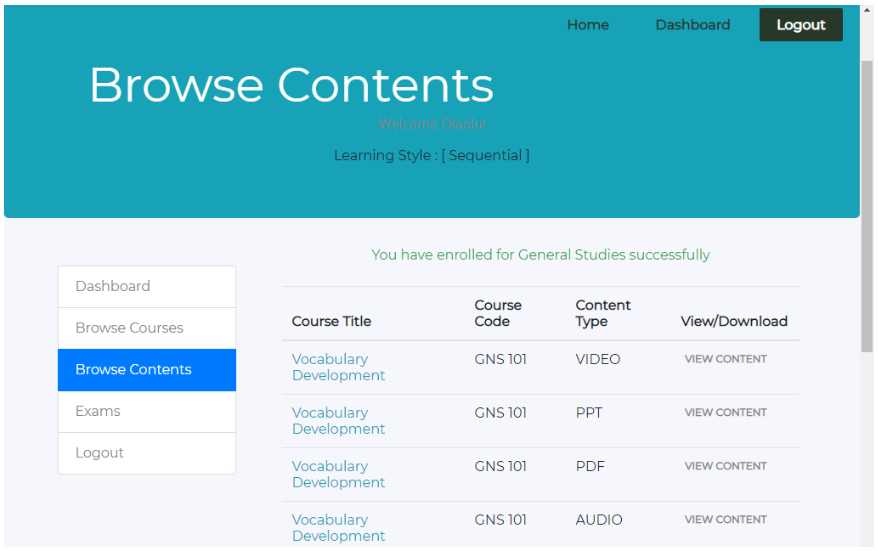
Course Name : General Studies

Course Code : GNS

Level of Proficiency: Intermediate

Kindly provide accuurate answers to the few questions below to validate your proficiency level

Fig. 9. Course Enrolment


Home Dashboard Logout

Browse Contents

Welcome Olaolu

Learning Style : [Sequential]

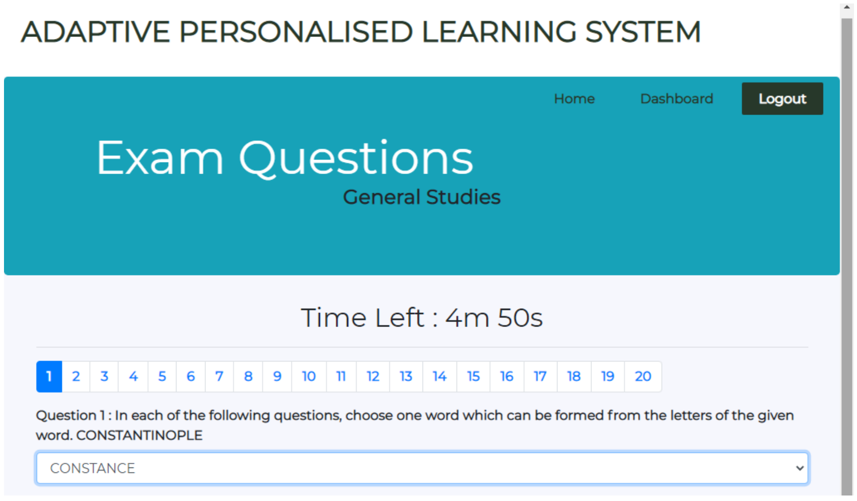
Dashboard
Browse Courses
Browse Contents
Exams
Logout

You have enrolled for General Studies successfully

Course Title	Course Code	Content Type	View/Download
Vocabulary Development	GNS 101	VIDEO	VIEW CONTENT
Vocabulary Development	GNS 101	PPT	VIEW CONTENT
Vocabulary Development	GNS 101	PDF	VIEW CONTENT
Vocabulary Development	GNS 101	AUDIO	VIEW CONTENT

Fig.10. Showing Personalized Content for the Learner

Different types of contents will be delivered to the user based on the learning style of the user. On the successful registration of the user for the beginner level, the system delivers contents to the user based on the learning style, proficiency level, and the profile of the user stored in the user profile repository (Figure 10).


ADAPTIVE PERSONALISED LEARNING SYSTEM

Home Dashboard Logout

Exam Questions

General Studies

Time Left : 4m 50s

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

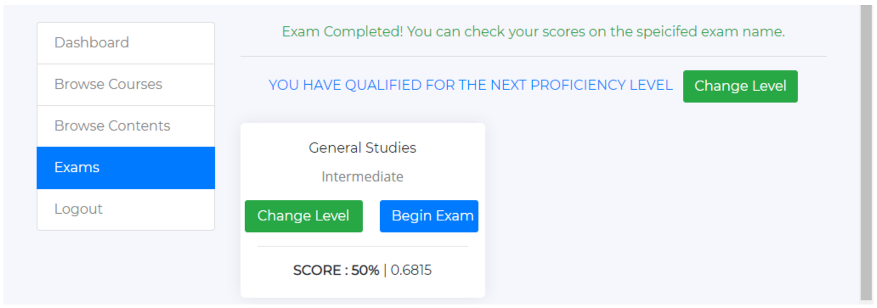
Question 1 : In each of the following questions, choose one word which can be formed from the letters of the given word. CONSTANTINOPLE

CONSTANCE

Fig. 11. Page Showing the Examination for the Beginner Level

The user is expected to view or download the various types of contents delivered to the user, after which the user is expected to take examination based on the content delivered (Figure 11).

The performance of the user in the examination taken, is the determinant of the eligibility of the user to move to the next level in the course. The course is slated for three-level, the beginner, the intermediate, and the expert level, such that the performance of the user at each level will show the eligibility of the user for the next level. The performance of the user reflects, the score of the user and the learning ability of the user (Figure 12).



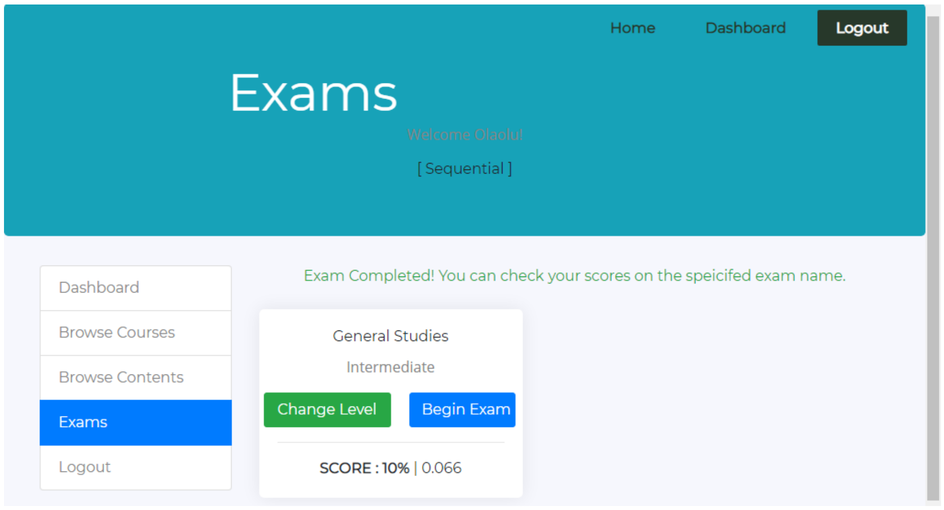
Exam Completed! You can check your scores on the speicifed exam name.

YOU HAVE QUALIFIED FOR THE NEXT PROFICIENCY LEVEL

SCORE : 50% | 0.6815

Fig. 12. Showing A Learner Score and Learning Ability Qualified for The Next Level

In Figure 12, the exam score is 50 percent and the learning ability of this user is 0.68 of 1, the system also communicates the eligibility of the user for the next level, but if the performance of the user score is not up to 50% for the next level (Figure 13); then the user cannot proceed to next level (Figure 14).



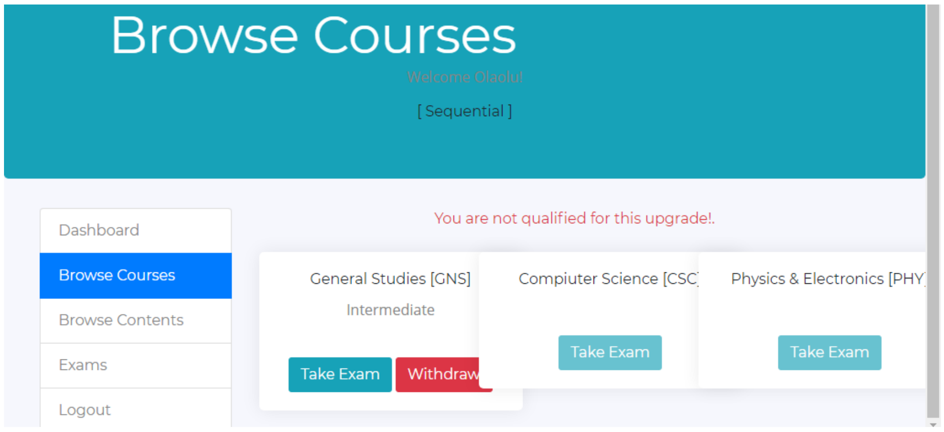
Exam Completed! You can check your scores on the speicifed exam name.

General Studies
Intermediate

SCORE : 10% | 0.066

Fig.13. Showing Examination Score and the Learning Ability

On the attempt to move to the next level, the system will display that the user is not eligible and will be taken back to the page, where the same level examination will be re-taken (Figure 14).



You are not qualified for this upgrade!.

General Studies [GNS]
Intermediate

Compiuter Science [CSC

Physics & Electronics [PHY

Fig. 14. Showing the System Result

5. EVALUATION

The course used for the case study is General Studies Course (GNS 101), an English course offered by all 100 level students of the Federal University of Technology, Akure, Nigeria. The system was designed for three categories of learners which include, the beginner, the intermediate, and the expert learners.

The learning contents were structures using ontology covering different categories. For the beginner category the contents include: I) Adjectives, II) Adverbs, III) Common Mistakes, IV) Comprehension, V) Direct and indirect Speeches while for the Intermediate Learner category of the general studies the contents include: I) Joining Phrase and Sentence II) Lexis And Structure, III) Noun And Pronouns, IV) Oral Forms, V) Prepositions and Contents for the Expert learners Category include: I) Punctuations Marks and their Uses, II) Spellings, III) Synonyms And Antonyms, IV) Verbs and Tenses, V) Word Combination.

The system was tested with twenty users, willing to respond to the conventional method of learning, so as to be able to carry out the comparative analysis of both methods. The mean performance value of the system was determined by obtaining the summation of the percentage score of all the users at each level divided by the number of users at each level (Table 1 and Figure 15).

Tab. 1. Showing comparison between the personalized system and the conventional system

S/N	PROFICIENCY LEVEL	MEAN PERFORMANCE VALUE
	Beginner	
1	Conventional method	52
2	Personalized adaptive	68.45
	Intermediate	
3	Conventional method	51
4	Personalized adaptive	64.9
	Expert	
5	Conventional method	51.1
6	Personalized adaptive	67.6

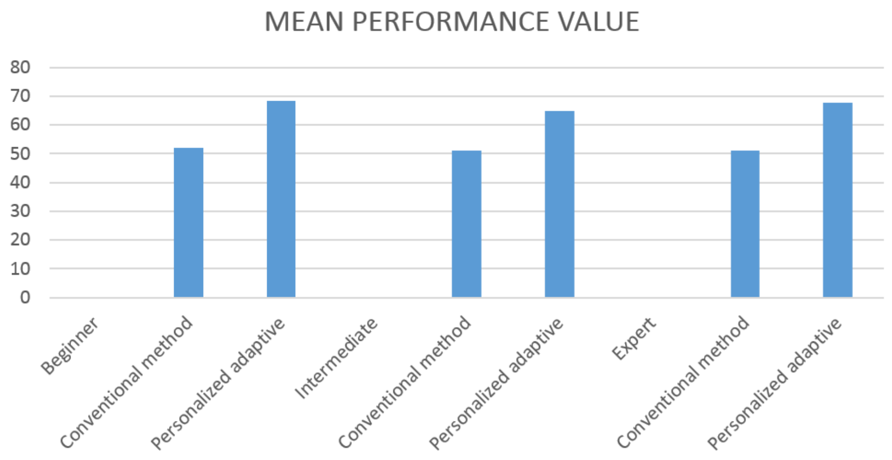


Fig. 15. Mean Performance Value

The system was evaluated with questionnaire filled by the twenty users of the system. The analysis is shown in Table 2 and Figure 16.

Tab. 2. Analysis Table

S/N	REMARKS	SATISFACTORY	GOOD	FAIR	POOR
1	System user-friendliness	10	6	4	0
2	System accuracy	11	7	2	0
3	System efficiency	9	6	3	2
4	System usability	18	1	1	0
5	System effectiveness	9	7	2	2

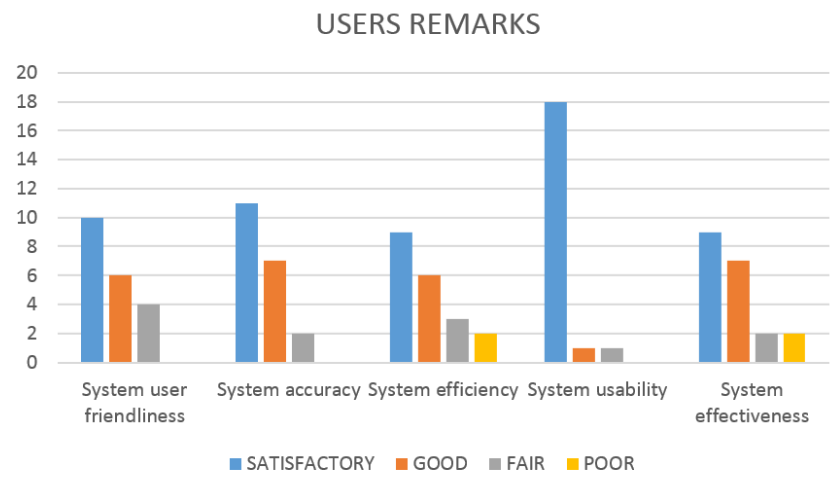


Fig. 16. Overall Performance of the System

Figure 16 shows the result of the evaluation in terms of user-friendliness, accuracy, efficiency, and the effectiveness of the system respectively. The result shows that the system is satisfactory as the majority of the users chose satisfactory as their remarks.

6. CONCLUSION

The personalization and adaptability of a system have been a technique, that has benefited the e-learning environment. However, in most existing personalized adaptive systems, learning contents are not tailored to the learners based on their learning styles. An ontology-based personalized adaptive e-learning system has been developed to offer a variety of personalized learning contents suitable to learners according to their learning styles. This will enhance their learning rate as it increases their learning abilities.

The system allows learners to take a learning style detector test to capture the learner's learning style but in the conventional e-learning system; the learning styles are not captured. The system delivered contents to the learner based on their learning style captured and go through a g-test to capture the learning ability of learners on a particular course. The examination was conducted for each learner within a space of time to determine the performance of the learner and to track the improvement in the learner's learning ability. The personalized adaptive e-learning system was tested using a General Study Course (GNS) as the learning materials with 20 users.

The results from the two methods were compared and the personalized adaptive system has a higher mean performance value at every level than the conventional methods, indicating that the system is more efficient and most preferred to the conventional method. Furthermore, the system was evaluated by 20 users in terms of System user-friendliness, System accuracy, System efficiency, System usability, System effectiveness. It was observed that a higher percentage of the users' remarks fall between satisfactory and good, which shows that the system was acceptable to them.

REFERENCES

Abu-Sayf, F.K. (1979). The Scoring of Multiple-choice Tests: A Closer Look. *Educational Technology*, *19*(6), 5–15.

Adewale, O.S. (2006). *University Digital libraries: an initiative for teaching, research, and service.* Adeyemo Publishing House.

Agbonifo, O., & Obolo, O. (2018). Genetic Algorithm-based Curriculum Sequencing Model for Personalized E-Learning System. *I.J. Modern Education and Computer Science*, *5*, 27–35.

Baker, F. (2001). *The Basics of Item Response Theory*. University of Maryland, College Park, MD: ERIC Clearinghouse on Assessment and Evaluation.

Baker, F.B. (1992). *Item Response Theory: Parameter estimation techniques*. Marcel Dekker.

Beulah, C., Latha, C.B., & Kirubakaran, E. (2013). Personalized Learning Path Delivery in Web based Educational Systems using a Graph Theory based Approach. *Computer Science*, 55428839.

Boyinbode, O., & Akintade, F. (2015). A Cloud Based Mobile Learning Interface. *Lecture Notes in Engineering and Computer Science: Proceedings of The World Congress on Engineering and Computer Science* (pp. 353–356). San Francisco, USA.

Boyinbode, O., & Bagula, A. (2012). An Interactive Mobile Learning System for Enhancing Learning in Higher Education. *Proceedings of the IADIS International Mobile Learning Conference Berlin* (pp. 331–334). Germany.

Brusilovsky, P. (2001). Adaptive hypermedia. *User Modeling, and User–Adapted Interaction*, *11*, 87-110.

Chen, C.M., & Chung, C.J. (2008). Personalized mobile English vocabulary learning system based on item response theory and learning memory cycle. *Computers & Education*, *51*(2), 624–647.

Chi,Y. (2009). Ontology-based Curriculum Content Sequencing System with Semantic Rules. *Expert Systems with Applications*, *36*(4), 7838–7847. https://doi.org/10.1016/j.eswa.2008.11.048

Crocker, L., & Algina, J. (1986). *Introduction to classical and modern test theory*. Holt, Rinehart, and Winston.

Dag, F., & Gecer, A. (2009). Relations between online learning and learning styles. *Procedia Social and Behavioral Sciences*, *1*, 862–871.

Hambleton, R.K., Swaminathan, H., & Rogers, H.J. (1991). *Fundamentals of Item Response Theory*. Sage Publications.

Hamzeh, M. (2005). Using Distractors in Correcting for Guessing in Multiple-Choice Tests. *Educational Sciences*, *32*(1), 192–197.

Kurilovas, E., Zilinskiene, I., & Dagiene, V. (2016). Recommending Suitable Learning Paths According to Learners' Preferences: Experimental Research Results. *Computers in Human Behavior*, *51*, 945–951.

Uhomoibhi, J.O. (2006). Implementing e-learning in Northern Ireland: prospects and challenges. *Campus-Wide Information Systems, 23*(1), 4–14.

Yarandi, M., Jahankhani, H., & Tawil., A. (2013). A Personalized Adaptive e-learning approach based on semantic web technology. *Computer Science*, 6257403.

Applied Computer Science, vol. 16, no. 4, pp. 85–102
doi:10.23743/acs-2020-31

Submitted: 2020-10-25
Revised: 2020-12-21
Accepted: 2020-12-28

Computer vision; Raspberry Pi system

Mohanad ABDULHAMID[*], *Otieno ODONDI*[**],
Muaayed AL-RAWI[***]

COMPUTER VISION BASED ON RASPBERRY PI SYSTEM

Abstract

The paper focused on designing and developing a Raspberry Pi based system employing a camera which is able to detect and count objects within a target area. Python was the programming language of choice for this work. This is because it is a very powerful language, and it is compatible with the Pi. Besides, it lends itself to rapid application development and there are online communities that program Raspberry Pi computer using Python. The results show that the implemented system was able to detect different kinds of objects in a given image. The number of objects were also generated displayed by the system. Also the results show an average efficiency of 90.206 % was determined. The system is therefore seen to be highly reliable.

1. INTRODUCTION

Object counting is an important image processing technique that is applicable in many industrial applications. Some examples of these applications include: counting the number of products passing a conveyor belt, counting the number of cars passing through a given road at a given time, or counting the number of a particular species in a wildlife park.

Cameras have become a standard hardware and a required feature in many mobile devices. These developments have moved computer vision from a niche tool to an increasingly common tool for very many applications such as facial recognition programs, gaming interfaces, industrial automation, biometrics, medical image analysis, and planetary exploration.

[*] Al-Hikma University, Karada Kharidge, Baghdad, Iraq, moh1hamid@yahoo.com
[**] University of Nairobi, P.O.Box 30197, GPO, Nairobi, Kenya, researcher12018@yahoo.com
[***] AL-Mustansiryia University, Baghdad, Iraq, muaayed@yahoo.com

Raspberry Pi is one such mobile device that comes with a built in camera slot. There are a number of applications that can be achieved through the Pi camera. Hobbyist use it to develop gaming programs and robotic applications. They direct a robot using a given set of image instructions such as, turn left, or right, or stop.

Computer vision is the automated extraction of information from images. Such information includes: 3D models, object detection and recognition, grouping and searching information, image warping, de-noising among others. Intelligent Transportation Society of America (ITSA) defines computer vision as the process of using an image sensor to capture images, then using a computer processor to analyze these images to extract information of interest.

Computer vision is used in a wide variety of real-world applications such as Optical Character Recognition (OCR) to read handwritten postal codes, rapid machine parts inspection in production plants for quality assurance, using stereo vision with specialized illumination to measure tolerances on aircraft wings or auto body parts, and looking defects in steel castings using X-ray vision. Computer vision is also used in object recognition for automated check points in retail, automotive safety by detecting unexpected obstacles such as pedestrians on the street, under conditions where active vision techniques such as radar do not work, and in medical imaging.

Raspberry Pi is defined as a low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is presented as a little device that enables people to experience computing and learn programming languages such as Scratch and Python. Essentially, it can perform anything that one would expect a desktop computer or laptop to perform. Some works which use Raspberry Pi in computer vision are found in literatures (Islam, Azad, Alam & Hassan, 2014; Jana & Borkar, 2017; Nikam, Doddamani, Deshpande & Manjramkar, 2017; Odondi, 2016; Sandin, 2017; Senthilkumar, Gopalakrishnan & Sathish Kumar, 2014).

2. DESIGN PROCEDURE

2.1. Hardware Requirements

This work is accomplished using the following hardware components: Raspberry Pi, Pi camera, and power supply.

2.1.1. Raspberry Pi and SD card

The design of this work uses PI (Model B+). In order to efficiently execute the work, Raspbian Jessie OS was installed in a 16GB Secure Digital (SD) card. As opposed to Raspbian Jessie Lite OS and Whizzy OS, Raspbian Jessie gives a Graphical User Interface(GUI) experience. Therefore, it was not imperative to use Putty to access the Raspberry Pi remotely. With Remote Desktop Protocol (xRDP) installed in the Pi, one can connect to the Raspberry Pi remotely using the Windows Remote Desktop Connection application. It was developed by Raspberry Pi foundation in UK to be used for the advancement of computer science education. The second version of the Raspberry Pi is used here.

2.1.2. Raspberry Pi Camera

The Raspberry Pi camera board plugs directly into the Camera Serial Interface (CSI) connector on the Raspberry Pi. The Raspberry Pi camera module attaches to Raspberry Pi by way of a 15 pin Ribbon cable to the dedicated 15-pin Mobile Industry Processor Interface (MIPI) CSI which was designed especially for interfacing to cameras. It is able to deliver a clear 5 megapixel resolution image or 1080p High-definition (HD) video recording at 30 frames/sec.

2.1.3. Power Supply

The power supply on Raspberry Pi is quite simple. It powers through a Micro Universal Serial Bus (USB) connection which is capable of supplying at least 700 mA at 5 V.

2.2. Ethernet Cable

There are various ways of accessing the Raspberry Pi. It is not possible to work on the Raspberry Pi on its own as it does not have either a monitor or a keyboard. It is therefore important to have an Audio-visual/High-Definition Multimedia Interface (AV/HDMI) display and a keyboard. However, it can also be accessed remotely by connecting it to a laptop or a desktop using an Ethernet cable. The later method was adopted for its convenience.

2.3. Software Used

The software used to successfully realize the objectives of the work include: Python, Open source Computer Vision software (OpenCV), Microsoft Office Visio and Word, Raspbian Jessie OS and Remote Desktop Connection application in Microsoft Windows 8.1 which granted remote access to Pi.

2.4. Modeling

The overall design is visualized at hardware level by block diagram of Fig. 1 while the flowchart of Fig. 2 gives the steps used in implementing the system.

2.4.1. Block Diagram

The integration of different hardware components is as shown by the block diagram of Fig. 1. The figure shows the integration of various hardware components.

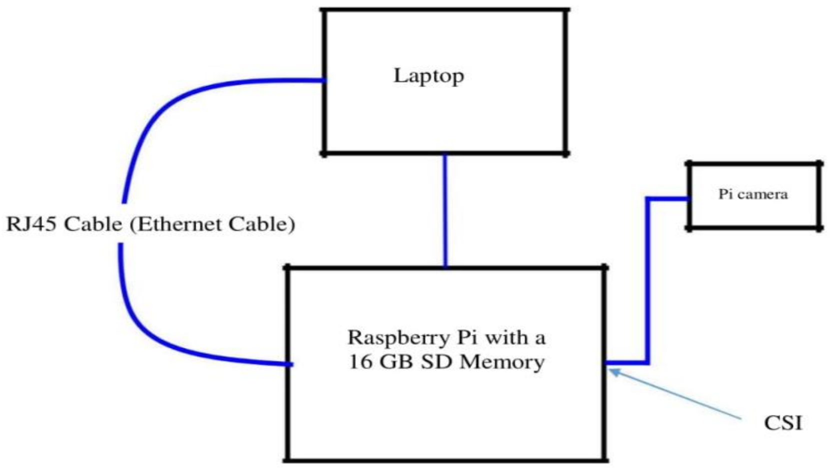

Laptop
RJ45 Cable (Ethernet Cable)
Pi camera
Raspberry Pi with a 16 GB SD Memory
CSI

Fig. 1. Block diagram showing the integration of the hardware

The Pi camera was inserted into the CSI slot provided on the Pi, an Ethernet cable is connected to the Ethernet ports of both Pi and the laptop to access Pi remotely. The micro USB 5 V 700 mA was used to power Raspberry Pi.

2.4.2. Methodology

There are several ways in which this task could be achieved. However, design settled on the solution shown in the flowchart of Fig. 2 below. The critical steps in the design were as outlined.

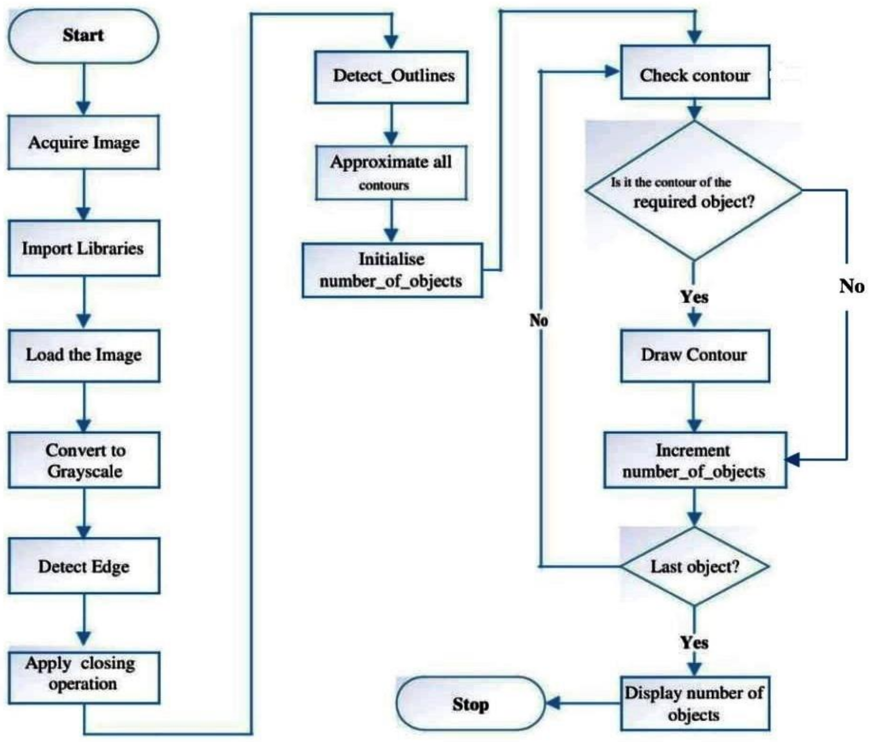


Fig. 2. Flow chart of the algorithm

2.4.2.1. Image Acquisition

This step involves image capture using the Pi camera mounted on the Raspberry Pi computer and saving it to Pi root directory. There are several ways of capturing an image using a Pi camera for example the command "*raspistill -o my_image.jpg*" captures and saves a Joint Photographic Experts Group (JPEG) image as *my_image* to the root directory of the Raspberry Pi.

2.4.2.2. Image Processing

The first step in image processing is to import the necessary OpenCV libraries (numpy and cv2) before load the image off disk using *cv2.imready ("image.jpg")* function. This function reads the image that was captured so that it can be processed. The next step is grayscale conversion. This is achieved using the *cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)* where *cv2.COLOR_BGR2GRAY* is the flag which converts a BGR image to a GRAY image.

It is necessary to blur the grayscale image slightly to reduce high frequency (Gaussian) noise and increase the efficiency of the algorithm. This was realised using *cv2.GaussianBlur(gray, (3, 3), 0).* The function takes four arguments. The first is the image, the second shown in the bracket are the height and the width of the standard deviation. Finally, we also need to specify sigmaX and sigmaY. If only sigmaX is specied, sigmaY is assumed to be the same as sigmaX. In this case, sigmaX was zero which implied that the two sigmas were calculated from the kernel. There are other functions in openCV that can also be used to blur an image such as averaging (*cv2.blur()*), median blurring (*cv2.medianBlur()*) and bilateral filtering (*cv2.bilateralFilter()*) but Gaussian blurring was used because it is highly effective in removing Gaussian noise from the image.

The next step is to detect the edges of objects in the image. Canny edge detection algorithm was used because it is widely accepted as the optimal detection algorithm. The openCV function *cv2.Canny(gray, 10, 250)* was used with 10 and 250 as the minimum and maximum values respectively.

In many cases, some outlines of the objects in the images are not "clean" and complete. There are usually gaps in between the outlines that must be closed if objects were to be successfully detected. To solve this, a "closing" operation was applied to close the gaps between the white pixels in the image. Two morphological transformation functions: *cv2.getStructuringElement(),* and *cv2.morphologyEx()* were used. *cv2.morphologyEx()* takes three inputs. The first one is the image while second is the morphological function, for example, cv2.MORPH_OPEN, or cv2.MORPH_GRADIENT, or cv2.MORPH_CLOSE, etc. depending on what we want to achieve. In this case cv2.MORPH_CLOSE is the argument passed since there is need to close the small holes inside the foreground objects, or small black points on the object. Lastly, the third input is by default kernel. On the other hand *getStructuringElement()* takes two arguments – the first one is dependent on the shape of kernel required, e.g. a rectangle or a cross, while the second argument specifies the matrix e.g. (7,7) for a 7×7 matrix.

The next step was to detect contours or outlines of the objects in the image. A contour is simply a curve joining all the continuous points (along the boundary), having same colour or intensity. *cv2.findContours()* function is used. It takes three inputs: the first one is the image, the second one is the contour retrieval mode and the third is one is the contour approximation method. The argument cv2.CHAIN_APPROX_SIMPLE is preferred to cv2.CHAIN_APPROX_NONE since the latter saves all the points of the contour while the former only save the end points of the contour thus saves on memory space. For a rectangle, only four points are saved.

To check if a contour is of the required object or not, it is necessary to loop over each of the contours one by one. The functions *cv2.arcLength(c, True),* where the first argument is the contour, while the second argument specifies whether the shape is a closed contour (if passed True) or just a curve, and *cv2.approxPolyDP (c, 0.02 * peri, True)* were used. The first argument passed is the counter and the next is epsilon which is usually a percentage of the arc length (‘*arcLength’)*. The third argument is as in the case of *arcLength()* function. Finally, the processed image is displayed on the screen using the *drawContours()* function.

2.4.3. Assumptions

The code assumed that a circle is a polygon with more than several edges. Any circular object in any image was therefore processed and displayed as an object with more than four edges.

3. RESULTS

3.1. Single Object

3.1.1. Rectangular Object

Figs. 3 to 7 show the steps involved in detecting and counting rectangular object. Fig. 3 shows the raw image captured while Fug.4 shows the same image in gray scale.

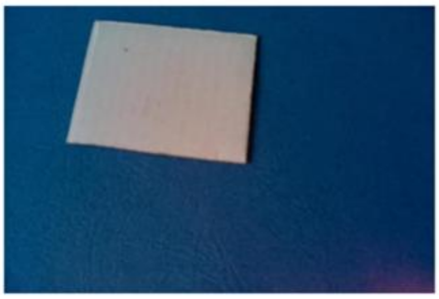


Fig. 3. Rectangular image

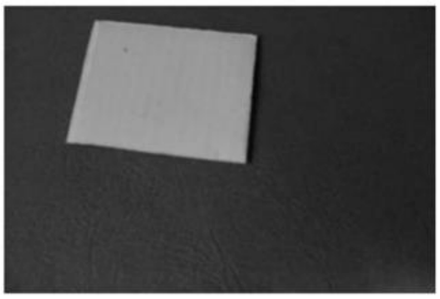


Fig. 4. Grayscale Image of Fig. 3

Fig. 5 shows the result of canny edge detection while Fig. 6 is the result of closing operation. While a clear cut difference between them may not be easily detected by the human eye, a keen look at the vertex of Fig. 6 (pointed to by an arrow) shows that a tiny gap at the same corner of Fig. 5 was closed. Latter results will clearly show this. Finally, Fig. 7 shows the output image displayed after image processing.

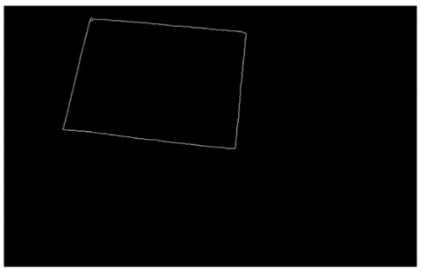


Fig. 5. Result of edge detection

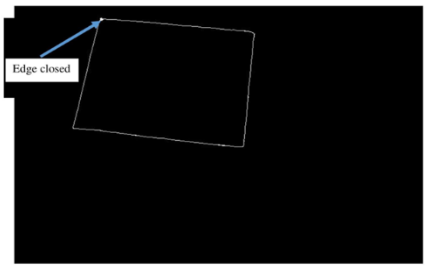


Fig. 6. Closed form of Fig. 3

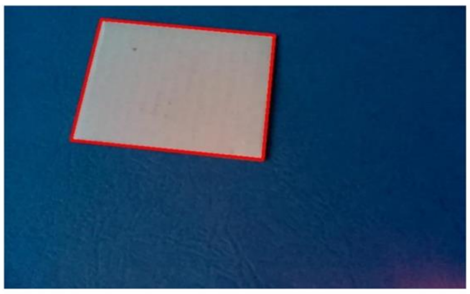


Fig. 6. Output of Fig. 3

In addition to the output, the code also retuned of the number of objects detected on the terminal upon successful completion. In this case, the following result was generated by the system:

There isn't any triangular object in the image.
There is 1 rectangular object in the image.
There isn't any object with more than four edges in the image.
I am glad to let you know that there is 1 object in this image.

3.1.2.Triangular Object

Similar observations were made as in the case of the one triangular object. Fig. 8 shows the test image while Fig. 9 is the blurred grayscale image of Fig. 8. The feedback from the terminal specifies that only one triangular object has been identified. Again there is a small gap as shown in Fig. 10 which is closed before the contour is drawn. Fig. 11 is the result of closing operation while Fig. 12 shows the output image displayed after image processing.

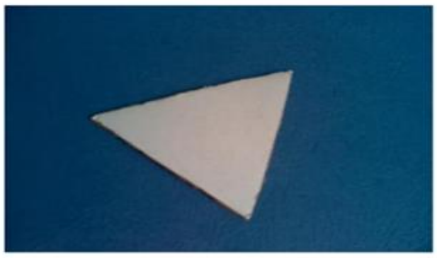


Fig. 8. Triangular object

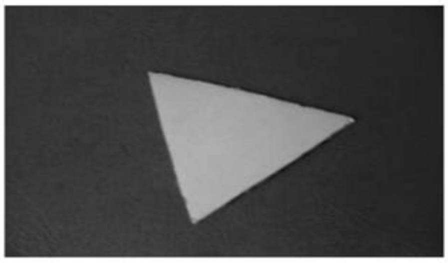


Fig. 9. Grayscale image of Fig. 8

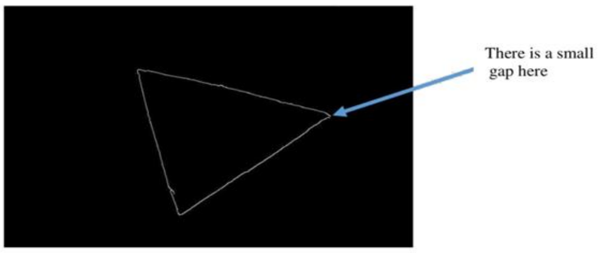


Fig. 10. Edged image

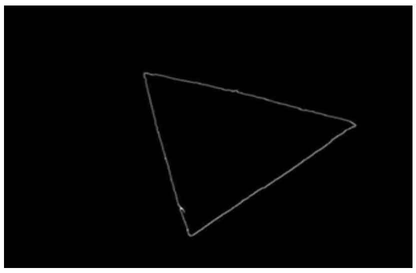


Fig. 11. Closed image

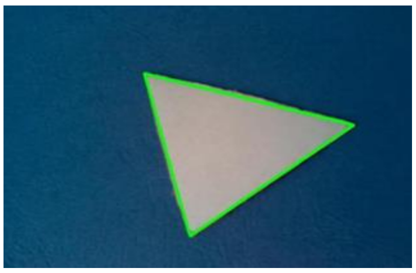


Fig. 12. System generated output

The system generated the following feedback when the code was executed to analyze Fig. 8 The result shows that only one triangular object was detected.

```
There is 1 triangular object in the image.
There isn't any rectangular object in the image.
There isn't any object with more than four edges in the image.
I am glad to let you know that there is 1 object in this image.
```

3.2. Multiple Objects

3.2.1. Human Vision versus Computer Vision

The test data of Fig. 13 was used to evaluate the performance of the code when there are different kinds of objects. While there are six rectangular shaped objects by human, the code only identified four of them. Other objects were accurately identified.

An interesting observation is noted here. Since the objects were cut without a ruler and by using a scissor i.e. freehand, the code was able to detect an extra vertex, which a human eye would not have detected at the bottom right corner of the image. This object is therefore identified as a polygon as shown in Fig. 17. The high level of efficiency is attributed to the fact that computer vision is based on evaluation at pixel level while the human vision is usually based on pre-recorded information. It is also necessary to note that the code classifies coins as polygons with more than four edges. This is because any circular shaped object seen as a polygon.

The object pointed to by the arrow in Fig. 13 was not detected by the code. The only parameter that stands out for this object is its small size relative to the other "human perceived rectangles". This is a typical error that would make human vision preferred to computer vision.

Fig. 14 shows the result obtained when Fig. 13 was converted to a grayscale image and blurring it. Fig. 15 is the result of Canny edge detection algorithm, while Fig. 16 is the result of closing and clearly shows the need for performing closing to an image before determining the type of the image. After edge detection, the twenty shillings coin (the coin on the extreme left of the test image – Fig. 13) is seen to have some gaps within its contours which without performing the closing operation could have been considered as objects within the coin.

The following result was generated by the system when the command *python countdiffobjects.py* was run in the terminal to process Fig. 13.

```
There are 2 triangular objects in the image.
There are 4 rectangular objects in the image.
There are 6 objects with more than four edges in the image.
I am glad to let you know that there are 12 objects in this image.
```

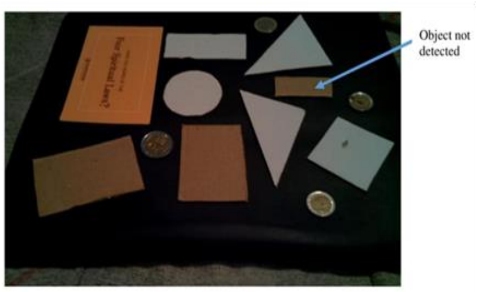


Fig. 13. Image with several objects

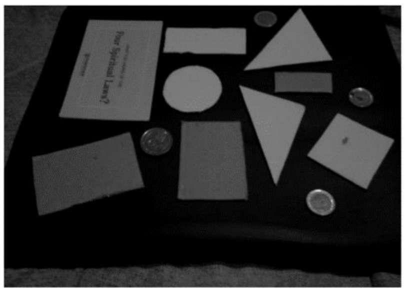


Fig. 14. Gray scale image

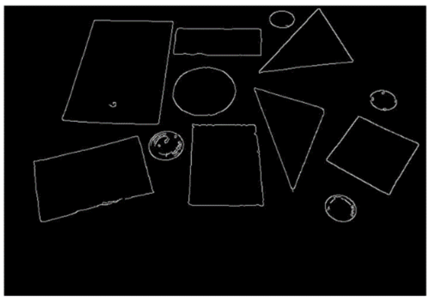


Fig. 15. Result of edge detection on Fig. 13

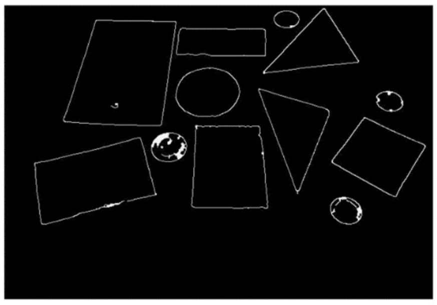


Fig. 16. Result of 'Closing' operation

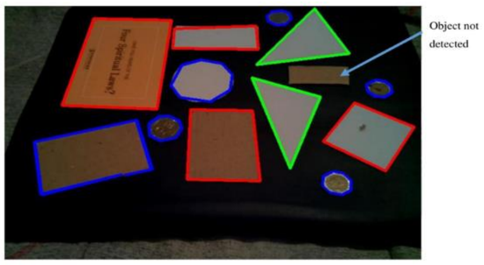


Fig. 17. Output of Fig. 13

3.2.2. Optimal Performance

The success of the code depended on the inter-object spacing and the distance from the object to the camera lens (focus). The code was found to be most reliable with a better trade-off between inter-object spacing and focus. Besides, the intensity of the background color affected the performance of the code. Using the specimen of Fig. 18, which was captured when the distance between the objects was optimal and with a better focus, the best result was realized.

Fig. 18 is the object under test while Fig. 19 is its grayscale image. Fig. 20 and Fig. 21 are the edged and closed images respectively. It was observed that all objects were successfully detected because of the sharp contrast in the color of objects and that of the background.

When the image of Fig. 18 was processed, the output image of Fig. 22 was displayed. All the objects were detected and counted. The rectangular and triangular objects were each five while objects with more than four edges were seven. In total therefore, seventeen objects were identified.

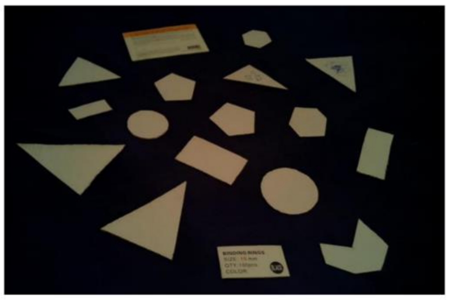


Fig. 18. Light colored objects in a black background

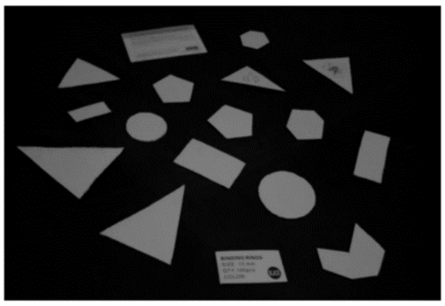


Fig. 19. Gray image of Fig. 18

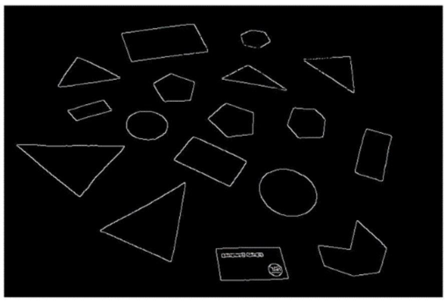


Fig. 20. Edged image of Fig. 18

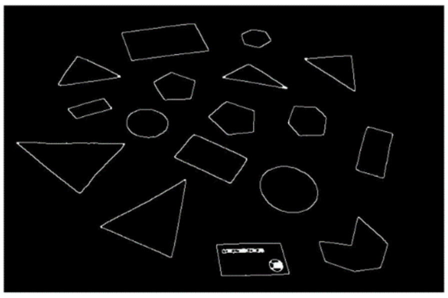


Fig. 21. Closed image of Fig. 18

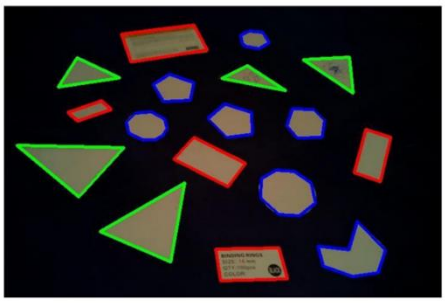


Fig. 22. Output of Fig. 18

The system generated the following results when Fig. 18 was processed:

There are 5 triangular objects in the image.
There are 5 rectangular objects in the image.
There are 7 objects with more than four edges in the image.
I am glad to let you know that there are 17 objects in this image.

The distance from the object to the camera is crucial in the performance of this code. As above figures show, the rectangular object at the bottom of those figures is ultimately detected as one object but when a close-up of the same image is taken, the words and the circular drawing within the same object are detected too.

3.3. Effect of Background Color and Inter-object Spacing on the System

The accuracy of the code was seen to reduce considerably when the objects were close to each other. This section illustrates how inter-object spacing and color intensity of the background in relation to that of the object affected the reliability of the code.

Fig. 23 was used to illustrate the two phenomena. It was observed that the spacing of the object, especially when there is no sharp contrast between background color and the color of object, two or more objects were identified as one.

When the image is converted to grayscale, the two pentagons pointed to by the red arrow in Fig. 23, were seen to be of almost the same intensity as the background (see Fig. 24). Their outline were therefore not detected as shown in Fig. 25 and Fig. 26. Equally, objects that are close to one another were also detected as one object as shown in Fig. 27. Since the rectangular object and the triangular object adjacent to it (pointed to by the blue arrow in Fig. 23) are very close to each other, their outlines were jointly detected as a single object (see Fig. 27). This is attributed to the fact that the color intensity of the images and background are almost the same in addition to them being close to each other.

The image was taken when the Pi camera was very closer to the object than the previous one. As such, the texture of the background affected the accuracy of the system as shown by the detection of non-object edges as shown in Fig. 25. Two triangles on the extreme left side, and the left-most rectangular object (Fig. 23) were therefore detected as one object as shown in Fig. 27.

The following result was obtained from the system when Fig. 23 was analyzed.

There are 2 triangular objects in the image.
There are 2 rectangular objects in the image.
There are 6 objects with more than four edges in the image.
I am glad to let you know that there are 10 objects in this image.

The result shows that only 10 objects were detected. In reality there were 17 objects as identified by human eye in the same image. The result shows the limitations of the system.

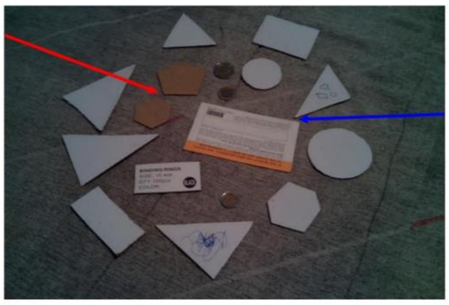


Fig. 23. Image with objects close to one another

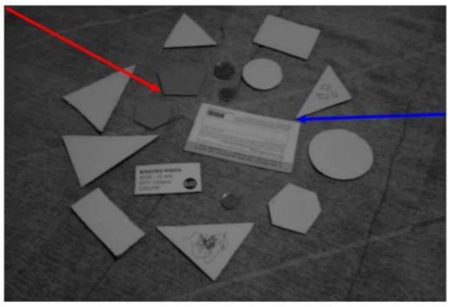


Fig. 24. Grayscale image of Fig. 23

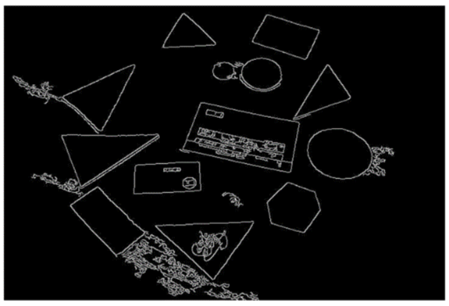


Fig. 25. Edged image of Fig. 23

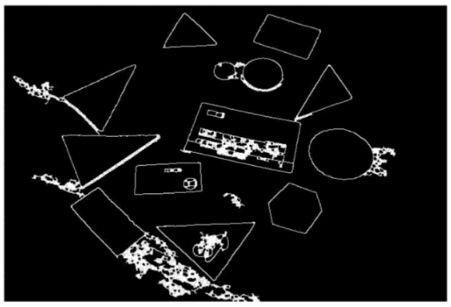


Fig. 26. Closed image of Fig. 23

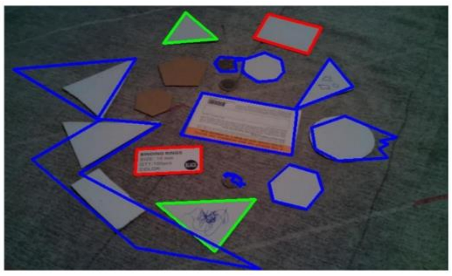


Fig. 27. Output of Fig. 23

3.4. Effect of Shadow on Code

The objects with shadow in the image could not be detected especially if there was no sharp contrast in color intensity with respect with the background. When the image of Fig. 28 was analyzed to detect rectangular shaped objects, errors were noted. This further showed other limitations of the code.

The rectangular box object had a shadow as shown in both Fig. 28 and Fig. 29 with a blue arrow. When Canny edge detector algorithm was applied, the edge with shadow was not detected. Besides, the eraser and the coin were not detected due to the effects of the background color. Fig. 30 and Fig. 31 show the edged image and closed image of Fig. 28 respectively.

As shown on the output generated by the system (Fig. 32), only one rectangular object is detected.

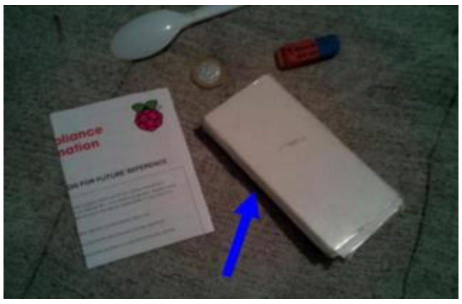


Fig. 28. Image with shadow

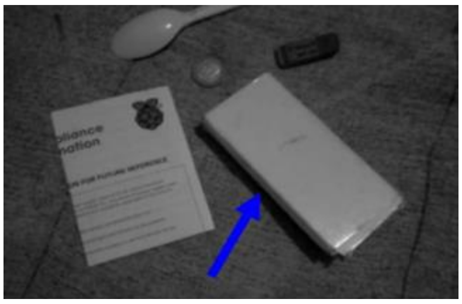


Fig. 29. Grayscale image of Fig. 28

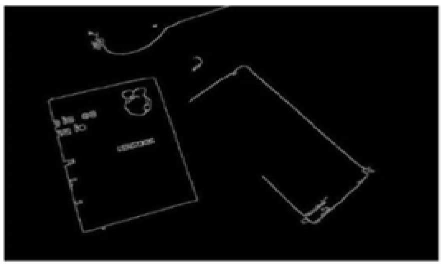


Fig. 30. Edged image of Fig. 28

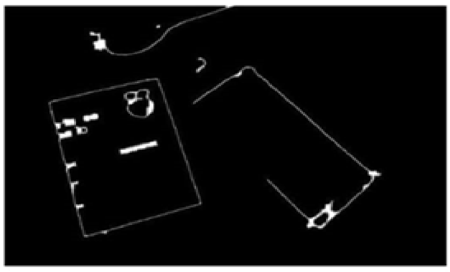


Fig. 31. Closed image of Fig. 28

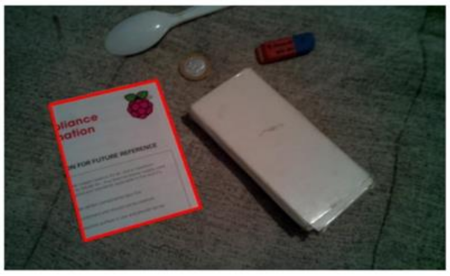


Fig. 32. Output of Fig. 23

3.5. Effect of Close-up Image on Code

The closing operation works perfectly as long as the gaps between the objects are small. When the gaps exceeds a threshold, the code identifies even the wittings on the objects. The following result was generated by the system when the code was executed for the image of Fig. 33. Fig. 34 shows the output image generated by the system.

There isn't any triangular object in the image.
There is 1 rectangular object in the image.
There are 32 objects with more than four edges in the image.
I am glad to let you know that there are 33 objects in this image.

By the human eye, this is a serious error since only one rectangular object was examined.

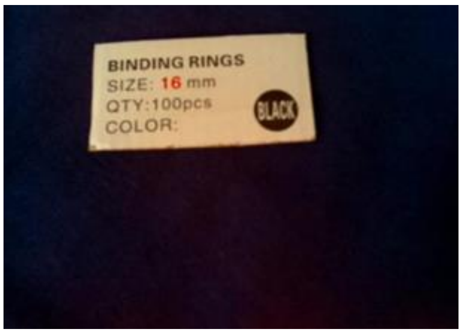


Fig. 33. Close-up image

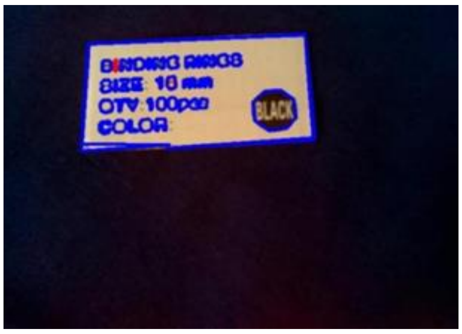


Fig. 34. Output when a close-up is taken

4. ANALYSIS

The functionality of the system is dependent on a number of factors which include: the background color in relation to the color of the object in the mage, the proximity of the camera to the object and by extension the texture of the background, the inter-object distance as well as the shadow as of the objects as was shown by the test results.

The test results for five objects – as listed in table 1 – were analyzed to determine the efficiency of the code.

Tab. 1. Efficiency analysis

Figure	**Number of Objects**	**Objects Detected by Pi Camera**	**Efficiency**
3	1	1	100%
8	1	1	100%
13	13	12	92.31%
18	17	17	100%
23	17	10	58.72%
		Total Efficiency:	**451.03**
		Average Efficiency:	**90.206**

From the table 1, the worst case scenario was experienced when Fig. 23 was analyzed. In this case, only 10 out of 17 objects were identified. The figures do not tell the whole story as revealed in Fig. 27 where it is clearly shown that only wrong objects were detected. This is because most objects were close to each other, the camera was closer to the object than other images that were analyzed and intensities of the background color and the objects were similar.

5. CONCLUSION

The work in this paper demonstrated the potential of the Raspberry Pi based system using Python as the programming language. The system was designed employing a camera which is able to detect and count objects within a target area. Different tests cases analyzed revealing the performance of the system. The reliability of the system was found out to depend on: number of objects within an image, the background color in relation to the color of the object, distance between objects, shadow of objects, and distance from the lens of the camera to the specimen (focus). The system was also able to differentiate between objects in an image based on their shapes. Rectangular objects, triangular objects and objects with more than four edges were easily detected. However, circular objects could not be detected as such partly because the camera was tilted at an angle and to a larger extent due to the assumptions outlined in the design section. The contours of circular detected as several edges with many vertices.

REFERENCES

Islam, M. M., Azad, M. S. U., Alam, M. A., & Hassan, A. (2014). Raspberry Pi and image processing based Electronic Voting Machine (EVM). *International Journal of Scientific and Engineering Research*, *5*(1), 1506–1510.

Jana, S., & Borkar, S. (2017). Autonomous object detection and tracking using Raspberry Pi. *International Journal of Engineering Science and Computing*, *7*(7), 14151–14155.

Nikam, A., Doddamani, A., Deshpande, D., & Manjramkar, S. (2017). Raspberry Pi Based obstacle avoiding robot. *International Research Journal of Engineering and Technology*, *4*(2), 917–919.

Odondi, O. (2016). *Computer Vision through the Raspberry-PI: Counting Objects* (graduation project). University of Nairobi, Kenya.

Sandin, V. (2017). *Object detection and analysis using computer vision* (graduation project). Chalmers University of Technology, Sweden.

Senthilkumar, G., Gopalakrishnan, K., & Sathish Kumar, V. (2014). Embedded image capturing system using Raspberry Pi system. *International Journal of Emerging Trends and Technology in Computer Science*, *3*(2), 213–215.

Applied Computer Science, vol. 16, no. 4, pp. 103–117
doi:10.23743/acs-2020-32

Submitted: 2020-11-30
Revised: 2020-12-07
Accepted: 2020-12-14

order violation, conflicts of resources, static analysis of the code

Damian GIEBAS[*], *Rafał WOJSZCZYK* [0000-0003-4305-7253][*]

ORDER VIOLATION IN MULTITHREADED APPLICATIONS AND ITS DETECTION IN STATIC CODE ANALYSIS PROCESS

Abstract

The subject presented in the paper concerns resource conflicts, which are the cause of order violation in multithreaded applications. The work focuses on developing conditions that can be implemented as a tool for allowing to detect these conflicts in the process of static code analysis. The research is based on known errors reported to developers of large applications such as Mozilla Firefox browser and MySQL relational database system. These errors could have been avoided by appropriate monitoring of the source code.

1. INTRODUCTION

The authors of some works concerning multithreading stress the need for diagnostic, monitoring or code optimization tools for developers, which will facilitate the so-called debugging process (Lu, Park, Seo & Zhou, 2008; Savage, Burrows, Nelson, Sobalvarro & Anderson, 1997). The basis for detecting such phenomena as race condition, deadlock, atomicity violation and order violation is the knowledge of resource conflicts which result in the mentioned phenomena. The conditions developed on the basis of resource conflicts research allow to carry out the process of static analysis of the source code to detect them (Giebas & Wojszczyk, 2020b; Lu et al., 2008; Park, Vuduc & Harrold, 2010). Phenomena such as race condition and deadlock are very well researched, and the literature contains many well documented methods allowing to locate the conflicts causing them (Bishop & Dilger, 1996; Cai, Wu & Chan, 2014; Giebas & Wojszczyk, 2018; Jin, Song, Zhang, Lu & Liblit, 2011; Netzer & Miller, 1992). Conflicts of

[*] Faculty of Electronics and Computer Science, Koszalin University of Technology, Śniadeckich 2,75-453 Koszalin, Poland, rafal.wojszczyk@tu.koszalin.pl

resources resulting in the phenomenon of atomicity violation are more complex than those concerning the previously mentioned phenomena, but there are also further successes in this field (Chew & Lie, 2010; Jin et al., 2011). The knowledge of resource conflicts causing a given phenomenon makes it possible to develop conditions allowing to analyse the code structure in order to detect them (Giebas & Wojszczyk, 2018, 2020a, 2020b, 2020c).

It turns out, however, that the atomicity violation, order violation and other undesirable phenomena can only occur in specific environments or on specific hardware configurations, as mentioned by Mozilla Firefox developers (Lu et al., 2008). Today, the multitude of combinations of settings, environments, and hardware configurations is so vast that it is impossible to perform enough tests in a real time to determine that the selected application is free of resource conflicts causing even one of the undesirable phenomena. As a result, applications are tested only on the most popular hardware platforms in environments based on the most popular operating systems. However, this process also has a number of disadvantages. Research conducted in 2017 showed that both developers and testers were usually unable to give the correct sequence of threads (Abbaspour Asadollah, Sundmark, Eldh & Hansson, 2017), i.e. knowledge of the scenario predicted by the architect or programmer implementing the indicated functionality is sometimes insignificant among other team members. In addition, the analysis of bug reports showed that the highest number of errors related to the phenomenon of order violation was classified in the Minor group, i.e. the fourth group on a scale from 1 to 5, where 5 are the least significant errors and 1 are the most significant ones. Therefore, the awareness of the threats posed by the phenomenon of order violation seems to be very low, which directly influences the amount of time spent on examining the causes of this phenomenon.

Data on the time needed to repair various types of errors were also analysed. The analysis shows that the repair of errors related to multithreading was 82 days on average, while the repair of errors not related to multithreading takes 66 days on average (Abbaspour Asadollah et al., 2017). This combined with the fact that very often the first modification of the code does not fix the error (Lu et al., 2008), it can be concluded that the average time spent by developers on fixing multithreaded errors is too short.

This work focuses on developing a condition for detecting resource conflicts that cause order violation. The element necessary for locating the searched conflicts turned out to be the sequential relations developed within the work (Giebas & Wojszczyk, 2020b).

A new definition of the phenomenon of order violation was developed as well. The own contribution should also include a review of actual errors in the open-source software and their analysis in order to develop conditions for locating resource conflicts causing the phenomenon of order violation. After the conditions have been developed, it is possible to implement the method as a computer program, used to code optimization.

The section after the introduction is a review of the state of knowledge in the field of multithreaded applications and the phenomenon of order violation. Section no. 4 describes research on known and well documented disorderly errors from Mozilla Firefox and MySQL relational database system, which is used in many software and scientific research (Abdulhamid & Kinyua, 2020). Section 5 formulates the problem and section 6 presents a sufficient condition. Section 7 discusses, among other things, the assumptions and limitations of the method developed. The discussion also includes the topic of checking whether the claim is true not only for the examples in section 4, but also for the order violation occurring in applications written in languages other than C language. It is worth noting that the C language is still very popular, and thanks to good optimization it is used in well-known single-board computers, e.g. Raspberry Pi (Cygan, Borowik & Borowik, 2018). Section 8 contains a leading example, where it is checked whether a simple example written in C is true. In the last section includes a summary of this work.

2. THE CURRENT KNOWLEDGE

An order violation is caused by reversing the order of access to two (or more) memory areas (i.e. A should always be invoked before B, but the order is not maintained during execution) (Lu et al., 2008). Thus, the application may be free of race condition, deadlock and atomicity violations, and yet its operation may be affected by irregularities.

This phenomenon has been classified to the group of phenomena of race character, as well as race condition and atomicity violation (Chen, Jiang, Xu, Ma & Lu, 2018; Torres, Marr, Gonzalez & Mössenböck, 2018; Lu et al., 2008). The character of the race should be understood as including time as one of the most important variables.

An example of such an application can be found in the order_violation_examples repository on the GitHub portal[*] in the order_violation.c file. Running this code several times may bring incorrect output in the console. This example is very simple, but it shows the essence of the problem. In order to eliminate the phenomenon of atomicity violation, 5 strategies have been proposed in the literature (Lu et al., 2008): control instructions, changing the order of operations, changing the source code structure, changing the position of operations assuming and releasing locks, and other solutions that do not fit into any of the previous groups.

The order violation in this example can be removed in two ways. In the first one, the whole loop should be placed in the critical section in function *t1f*. The second solution is to run the second thread after the first thread has finished

[*] https://github.com/PKPhdDG/order_violation_examples

working, which will ensure that the operation is launched in the right order. This example illustrates how complicated is the phenomenon of order violation.

The literature says that in one version of the Apache server code, the time needed to restore a order violation took 22 hours of uninterrupted server operation with an eight-core processor (Park et al., 2009). However, rarely does a single restoration of the phenomenon allow to understand and eliminate it. This example shows how much tools are needed to search for phenomena in real time.

One of solution is to use a different type of memory (Andrew, Mcpherson, Nagarajan, Sarkar & Cintra, 2015). The research shows that even the use of software transactional memory (STM) provided by Convoider software is not able to protect against the phenomenon (Yu, Zuo & Xiong, 2019). The authors of Convoider estimate that the use of transactional memories will allow to avoid order violation with a probability equal to 0.5%.

The phenomenon of order violation is also mentioned in the research on a testing technique called fuzzing. The ConFuzz tool, developed for the analysis of multithreaded applications, has been classified as a static code analysis tool (Vinesh & Sethumadhavan, 2020), because it reduces the application code to bitcode using the *llvm* compiler tools. The bitcode is then analysed. The results of the work do not contain any information about the location of conflicts causing the order violation, but the innovative approach may prove to be effective.

In the presented literature, it was not possible to find any clues or conditions allowing to locate resource conflicts causing order violation phenomena.

3. MODEL

In the following sections, Mozilla Firefox and MySQL source code fragments are also presented in graphical form, according to the source code model representation of a multithreaded application, which is as follows (Giebas & Wojszczyk, 2020b):

$$C_P = (T_P, U_P, R_P, O_P, Q_P, F_P, B_P) \tag{1}$$

where: P – the program index,

$T_P = \{t_i \,|\, i = 0 \ldots \alpha\}$, $(\alpha \in \mathrm{N})$ – a set of all threads of t_i application C_P, where t_0 is the main thread, $|T_P| > 1$,

$U_P = (ub \,|\, b = 1 \ldots \beta)$, $(\beta \in \mathrm{N}^+)$ – is the sequence of sets of ub, which are subsets of T_P containing threads working in the same period of time in the program C_P, whereas $|U_P| > 2$, $u_1 = \{t_0\}$ and $u_\beta = \{t_0\}$,

$R_P = \{r_c \,|\, c = 1 \ldots \gamma\}$, $r_c = \{v_1, v_2, \ldots, v_\eta\}$, $(\gamma, \eta \in \mathrm{N}^+)$ – a collection of shared application resources C_P, and the following elements are sets of variable names referring to a single resource,

$O_P = \{o_{i,j} \mid i = 1 \ldots \delta,\ j = 1 \ldots \epsilon\}$, $(\delta, \epsilon, \in \mathrm{N}^+)$ – is a set of all application operations of C_P, which at a certain level of abstraction are atomic operations, i.e. they cannot be divided into smaller operations; an operation is understood as an instruction or function defined in the programming language; an index i and indicates the number of the thread in which the operation is executed, and an index j is an order number of operations working within the same thread,

$Q_P = \{q_s \mid s = 1 \ldots \kappa\}$, $q_s = (w_s, x_s)$, $(\kappa, \in \mathrm{N}^+)$ – a set of all mutexes available in the program, defined as a pair variable, mutex type, where the type is understood as one of the set values (PMN, PME, PMR, PMD), where values correspond to the lock types in the library *pthread*,

$F_P = \{f_n \mid n = 1 \ldots \iota\}$ and $F \subseteq (O_P \times O_P) \cup (O_P \times R_P) \cup (R_P \times O_P) \cup (O_P \times Q_P) \cup (Q_P \times O_P)$, $(\iota \in \mathrm{N}+)$ – a set of edges including:

1. *Transition edges* – defining the order of operations. These edges are pairs $f_n = (o_{i,j}, o_{i,k})$, where the elements describe two consecutive operations $o_{i,j} \in O_P$,
2. *Usage edges* – indicating resources that change during the operation. These edges are pairs $f_n = (o_{i,j}, r_c)$, in which one element is operation $o_{i,j} \in O_P$, and the other is resource $r_c \in R_P$,
3. *Dependency edges* – indicating operations depending on the current value of one of the resources. These edges are pairs $f_n = (r_c, o_{i,j})$, where the first element is the resource $r_c \in R_P$, and the second is the operation $o_{i,j} \in O_P$,
4. *Locking edges* – indicating the operation applying the selected lock. These edges are pair $f_n = (q_s, o_{i,j})$, in which one element is the lock, and the other is the locking operation.
5. *Unlocking edge* – indicating the operation releasing the selected lock. These edges are pairs $f_n = (o_{i,j}, q_s)$, in which one element is the unlocking operation, and the other is the released lock.

$B_P = (B_P^{FWD}, B_P^{BWD}, B_P^{SYM})$ – set sequence:

B_P^{FWD} – set of pairs of **forward**-relationship operations: $B_P^{FWD} = \{(o_{i,j}, o_{a,b}); o_{i,j}, o_{a,b} \in O_P\}$; the first operation from the pair forces the second operation, while the second operation does not force the first. In the further part of the work it will be marked with the symbol $o_{i,j} \rightarrow o_{a,b}$,

B_P^{BWD} – a set of pairs of **backward** operations: $B_P^{BWD} = \{(o_{i,j}, o_{a,b}); o_{i,j}, o_{a,b} \in O_P\}$; the occurrence of the first operation from the pair does not force the second operation, while the occurrence of the second operation requires the first operation. In the further part of the work it will be marked with the symbol $o_{i,j} \leftarrow o_{a,b}$,

B_P^{SYM} – a set of pairs of **symmetric** relationship operations: $B_P^{SYM} = \{(o_{i,j}, o_{a,b}); o_{i,j}, o_{a,b} \in O_P\}$; the occurrence of the first operation from the pair forces the second one and conversely, the occurrence of the second operation from the pair requires the first one to occur. In the further part of this work it will be marked with the symbol $o_{i,j} \leftrightarrow o_{a,b}$.

An extension was introduced to the model consisting in changing the definition of a **symmetrical** relation. All symmetrical relations are a set of pairs of operations, because both operations must be performed in a given order, however, these operations can occur in two different threads. As a result, a two-element set consisting of operations of two different threads does not have information which of the operations should logically be performed first.

4. STUDIES ON THE ORDER VIOLATION

The review of the literature on the phenomenon of order violation did not bring the expected results in the form of conditions that the source code must meet in order for a resource conflict resulting in order violation to occur. The development of such conditions has already made it possible to locate the phenomena of race condition, deadlock and atomicity violation (Giebas & Wojszczyk, 2020a, 2020b, 2020c). The resource conflicts causing the order violation phenomenon should also have a number of common characteristics, which will enable locating them. In order to find these characteristics, it is necessary to analyse several fragments of the source code, the activation of which results in the phenomenon of order violation. Therefore, based on the literature, Mozilla Firefox and MySQL source code fragments will be reviewed, in which the resource conflicts bringing order violation will be analysed. All of these code fragments have been discussed in a paper (Lu et al., 2008), which generally discusses multithreaded application errors.

The file figure_2_mozilla_firefox.c, which is located in the **order_violation_examples** repository, contains an extract from Mozilla Firefox, the execution of which will result in the order violation. The application allows for this to happen when a thread using the *mMain* function will be run first and perform a dereference operation on the *mThread* resource, resulting in an unexpected termination of the application as a result of the order violation.

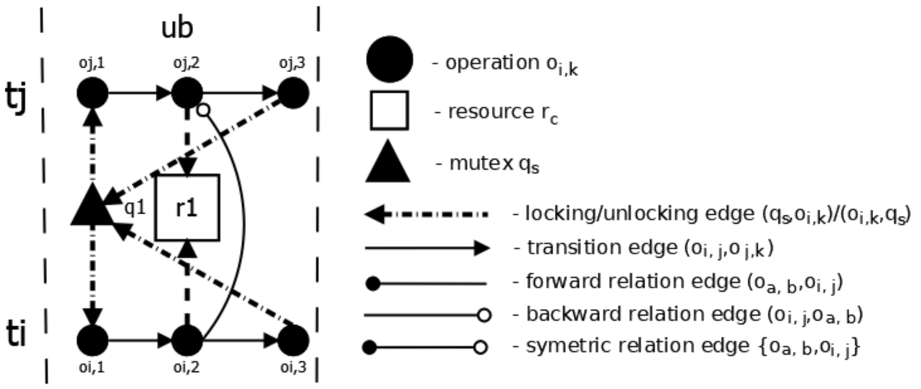

ub
tj
ti
oj,1
oj,2
oj,3
q1
r1
oi,1
oi,2
oi,3
- operation $o_{i,k}$
- resource r_c
- mutex q_s
- locking/unlocking edge $(q_s, o_{i,k})/(o_{i,k}, q_s)$
- transition edge $(o_{i,j}, o_{j,k})$
- forward relation edge $(o_{a,b}, o_{i,j})$
- backward relation edge $(o_{i,j}, o_{a,b})$
- symetric relation edge $\{o_{a,b}, o_{i,j}\}$

Fig. 1. File code figure_2_mozilla_firefox.c. as a graph

Thus, it will be true to say that there is a **backward** relationship (Giebas & Wojszczyk, 2020b) between the dereferencing operation and the initialization operation. This example shows that in Firefox application there are **backward** relationships between two operations of two different threads, and the reversed order of these operations with shared resource results in the phenomenon of order violation.

Another file from the **order_violation_examples** repository named figure_4_mozilla_firefox.c similarly to the previous one contains a piece of Mozilla Firefox browser code. The comment in the code shows that the second thread (and thus the *DoneWaiting* function) is launched at the end of the *PBReadAsync* function. As a result, one of the operations of the first thread is the reason for starting the second thread, with both operations changing the content of the *io_pending* resource in the same interval.

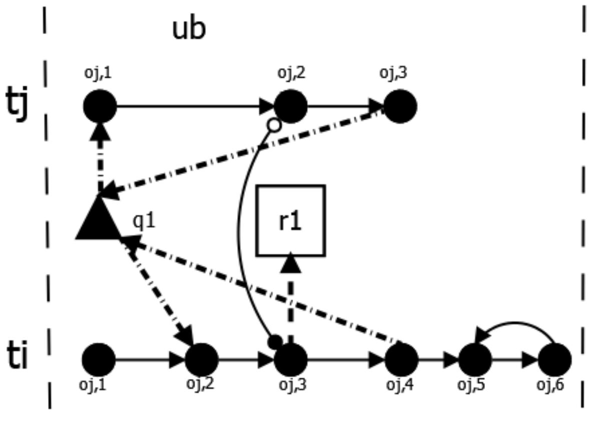


Fig. 2. File code figure_4_mozilla_firefox.c as a graph

From the description of the function contained in the article (Lu et al., 2008) it follows that first the resource should store the *TRUE* value and then *FALSE*. Therefore, it can be concluded that both value assignment operations are bound by a **symmetric** relation (Giebas & Wojszczyk, 2020b). The conflict has been resolved by moving the operation of assigning *TRUE* value to the resource *io_pending* over *PBReadAsync* operation. In the context of the proposed source code model of multithreaded applications, the repair was made by moving the operation to the previous time frame, so that it is certain that the *TRUE* value assignment operation will always be performed before the *FALSE* value assignment operation. Thus, as in the previous case, the resource conflict causing the order violation was the reversed order of execution of a pair of operations on a shared resource.

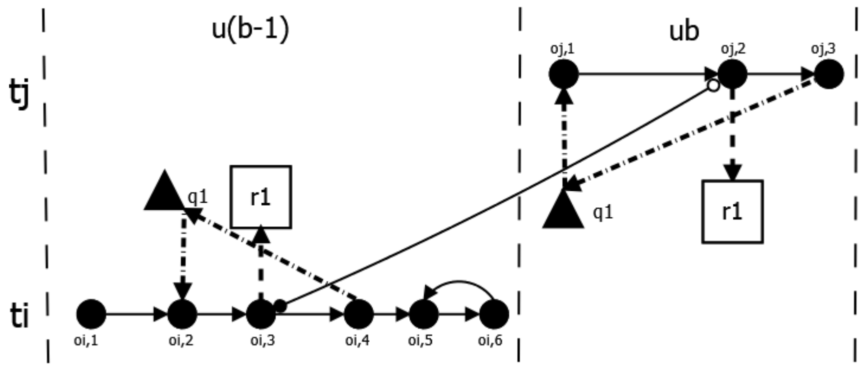


Fig. 3. Graph of the source code from the file figure_4_mozilla_firefox.c after taking into account modifications eliminating the resource conflict

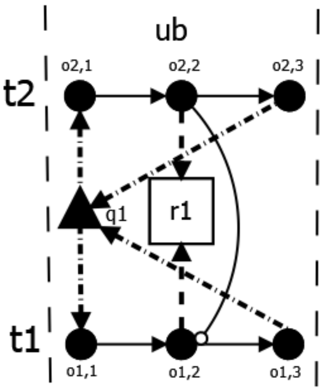


Fig. 4. Source code from the file figure_5_mozilla_firefox.c as a graph

The next piece of Mozilla Firefox browser code is in the file figure_5_mozilla_firefox.c of the aforementioned repository. In this case, it is the second thread operation that must be performed first. Every time *js_DestroyContext* is called, operations are performed on the shared *atoms* resource. The last time this function is executed by the first thread, the *js_UnpinPinnedAtom* function is performed, which executes the operation of freeing resources of the *atoms* variable. The result of this operation is unexpected termination of the browser operation, because in the second thread the *js_MarkAtom* function is called, whose parameter is the atoms variable with the value *nullptr*. This example is very similar to the previous two. The phenomenon of order violation occurs when the order of operations on the shared resource is reversed, which is the atoms variable. In this situation calling the *js_UnpinPinnedAtom* function cannot precede the *js_MarkAtom* function, so there is a **backward** relationship between them. The last piece of code comes from the MySQL database system and is in the figure_7_mysql.c file. In the first thread, the *dynamicId* variable is initialized, which is a shared resource. The handle for this resource is stored in the *dynamicId* variable of the *m_state* component of the node structural variable. Thus, if the second thread is run faster than the first thread, the uninitialized variable will be attempted to dereferencing, which in this case will lead to indefinite application behavior. As in the first example from Mozilla Firefox, there is a backward relationship between the two operations. The operations are performed in reverse order, with the result that a dereference is performed on an indicator variable for which memory has not been allocated, resulting in the order violation phenomenon.

The analysis of four resource conflicts resulting in the order violation, coming from large applications such as undoubtedly Mozilla Firefox browser and MySQL database system, has led to the following conclusions. The pairs of operations to which the definition of a violation of order refers should, according to the programmer’s assumptions, be performed in the order specified by a certain algorithm. It is from the algorithm that a logical order is derived, on the basis of which one of the three types of relations that may occur between the operations (Giebas & Wojszczyk, 2020b) is determined. The algorithm assumes that these operations will be performed in a specific order, so the relation connecting the two operations is a sequence relation and performing the operations contrary to this order results in a violation of the order.

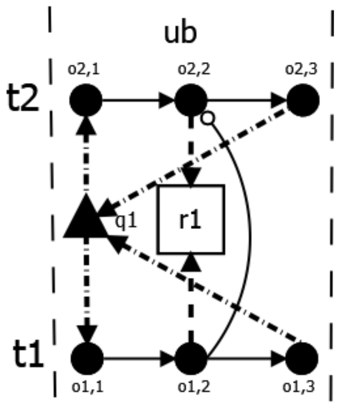


Fig. 5. Source code from the file figure_7_mysql.c as a graph

According to the current knowledge about resource conflicts causing the order violation phenomenon, the definition of this phenomenon is:

Definition 1. An order violation is a phenomenon where, between two operations of two different threads (or groups of operations), there is a sequential relationship whose reversal causes the algorithm to malfunction and an undefined state of the shared resources that have been used by the algorithm.

5. PROBLEM FORMULATING

The source code of the multithread application P is given, written in C using the *pthread* library. In this application there are sequential relations between operations of two threads and at least one pair of operations connected with the sequential relation is executed in the same time interval. This application is also free of race condition, deadlock and atomicity violation.

Therefore, is it possible to locate conflicts causing the phenomenon of order violation?

6. SUFFICIENT CONDITION

The source code model for multithreaded applications presented in section 3 will be used to develop a sufficient condition. Based on the examples presented in section 4, the statement of order violation is as follows:

Theorem 1. Let P be a multithreaded application free of race condition, deadlock and atomicity violation. So let $B_P = (B_P^{FWD}, B_P^{BWD}, B_P^{SYM})$ will be a set of pairs of operations which are in sequential relationship with each other, and ${}^{i,j}B_P^{\xi} \subseteq B_P^{\xi}$ will be a subset containing such pairs of operations $(o_{i,\alpha}, o_{j,\beta})$, the first of which is done in a thread ti and the second in the thread t_j.

If $\{t_i,t_j\} \subseteq u_b$ *then there will be a violation of order in the implementation of the operation* $(o_{i,\alpha},o_{j,\beta})$.

Proof. Proof is a direct consequence of the definition of a violation of order. If the threads $\{ti,tj\}$ are performed in a common interval of time, i.e. $\{t_i,t_j\} \subseteq u_b$ it is therefore acceptable to implement the concurrent operation $(o_{i,\alpha},o_{j,\beta})$. This means at the same time that any order of execution of the operation is possible, i.e.: $o_{i,\alpha} \rightarrow o_{j,\beta}$, $o_{i,\alpha} \leftarrow o_{j,\beta}$, $o_{i,\alpha} \leftrightarrow o_{j,\beta}$. It is therefore permissible to violate the set order of operations $(o_{i,\alpha},o_{j,\beta})$.

7. DISCUSSION

The definition of order violation from section no. 2 did not give any premises as to how to search for resource conflicts causing the discussed phenomenon in the source code of the application. Only the analysis of fragments of applications containing resource conflicts causing the phenomenon of order violation, taking into account the relations described in the paper (Giebas & Wojszczyk, 2020b), allowed for redefinition of the phenomenon and development of conditions allowing for detection of these conflicts, using the source code model of multi-threaded applications.

It can be stated with certainty that the detection of conflicts causing the phenomenon of order violation will be excessive, similarly as it is the case with the detection of conflicts causing race condition and atomicity violations (Giebas & Wojszczyk, 2020a, 2020b). In other words, the results will include the so-called false-positive error. It can also be stated that, despite the redundancy, it will be possible to ignore some conflicts with poorly defined relationships between the two operations.

It is also worthwhile to verify in the future the no. 1 definition based on source code of applications other than Mozilla Firefox and MySQL, and in which there is also a violation. The applications under study do not necessarily have to be written in C language. As soon as the application code manages to determine whether functions (or methods for languages supporting only object-oriented paradigm) are in one of the three developed relationships (Giebas & Wojszczyk, 2020b), and any shared resource is involved in the whole process, an attempt can be made to confirm this definition.

The statement of order violation from section 6 allows to locate the violation in all four cases described in section 4. In each of the described examples this phenomenon occurs because the structure allows to perform the operation contrary to the programmer's assumptions. According to the source code model of multithreaded applications, for two operations to be performed in a given order, the operations must belong to one thread. In a situation where both operations are in different threads, the order of execution can be forced only by placing operations in two different intervals. This type of solution has been used to eliminate the conflict causing atomicity violation in the second of the discussed examples in section no. 4. The graph presenting this solution can be found in figure no. 3.

8. LEADING EXAMPLE

Half of the examples described in sec. 4 concern the execution of an action on a resource before any memory resources are allocated to that resource. A common mistake in applications written in C by inexperienced programmers is to use indicator variables without checking the state of such variable first. In multithreaded applications it is additionally necessary to synchronize threads, so that the thread using indicator variable does not cause application failure. Such synchronization does not occur in OV1 application code located in motivation_example.c file in order_violation_examples repository. The first thread of this application is responsible for allocating space on the heap and returning the indicator to the indicator variable, and the second thread is responsible for copying to the address indicated by this indicator variable. The result of incorrect order of execution of the operation is unexpected termination of the application.

A common practice in writing multithreaded applications is to allocate memory in a different thread than other operations performed on it. In the *t2f* function of the leading example, just checking if the indicator variable does not indicate the NULL value and taking action only if this value is correct and it does not solve the problem. The programmer should ensure that the *memcpy* function receives an indicator to the allocated memory. This problem can be solved in several ways. The first way belongs to the group of naive solutions, i.e. the thread waits for the indicator to change its state by cyclic checking it in a loop, which can lead to waiting indefinitely. The second naive solution seems to be to sleep the thread for a given time by using the *sleep* function. In practice, this solution is worse than the previous one, because the time operation of the first thread is unknown, so the waiting time can be either overestimated or underestimated, and whether this value is overestimated or underestimated is strongly dependent on the hardware configuration on which the application will run. The only correct solution to this type of problem is to move the memory allocation operation with the thread to the previous time interval, as Mozilla developers have done by fixing one of the errors in Firefox.

The source code of the leading example in the model is as follows:
$T_{\mathrm{OV1}} = (t_0, \mathrm{t}_1, \mathrm{t}_2)$
$U_{\mathrm{OV1}} = (\{t_0\}, \{\mathrm{t}_1, \mathrm{t}_2\}, \{t_0\})$
$R_{\mathrm{OV1}} = \{(string)\}$

$O_{\mathrm{OV1}} = \{o_{0,1}, o_{0,2}, o_{0,3}, o_{0,4}, o_{0,5}, o_{0,6}, o_{1,1}, o_{1,2}, o_{1,3}, o_{1,4}, o_{2,1}, o_{2,2}, o_{2,3}, o_{2,4}, o_{2,5}, o_{2,6}\}$
$Q_{\mathrm{OV1}} = \{(n, PMD)\}$
$F_{\mathrm{OV1}} = \{(o_{0,1}, o_{0,2}), (o_{0,2}, o_{0,3}), (o_{0,3}, o_{0,4}), (o_{0,4}, o_{0,5}), (o_{0,5}, o_{0,6}), (q_1, o_{1,1}), (o_{1,1}, o_{1,2}), (o_{1,2}, r_1), (o_{1,2}, o_{1,3}), (o_{1,3}, q_1), (o_{1,3}, o_{1,4}), (o_{2,1}, o_{2,2}), (o_{2,2}, o_{2,3}), (q_1, o_{2,3}), (o_{2,3}, o_{2,4}), (o_{2,4}, r_1), (o_{2,4}, o_{2,5}), (o_{2,5}, q_1), (o_{2,4}, o_{2,6})\}$
$\mathrm{B}_{\mathrm{OV1}}^{\mathrm{SYM}} = \{(o_{1,2}, o_{0,5})\}$
$\mathrm{B}_{\mathrm{OV1}}^{\mathrm{BWD}} = \{(o_{1,2}, o_{2,4})\}$

Therefore, in order to locate the order violation phenomenon in the OV1 application, we must follow the theorem in section 6. Which means that the OV1 application includes a pair of operations $(o_{1,2}, o_{2,4})$, which is connected by a **backward** relationship and these operations belong to two different threads performed in the same time interval u_2. Both operations use a shared resource which is a *string* indicator variable. This means that the theorem is fulfilled, so there is a resource conflict in the application, which consists in reversing the order relationship resulting in the phenomenon of order violation.

9. SUMMARY

Based on actual errors and the current state of knowledge, a criterion has been developed in this work that can be implemented as an algorithm to locate resource conflicts in the process of static code analysis. However, the developed criterion is imprecise and may not include all real cases. On the other hand, the results obtained may be redundant, i.e. they may contain the so-called *false-positive error*. To a large extent, the location of resource conflicts that cause order violation is influenced by the correct definition of relations that may occur between operations.

Despite the disadvantages of static code analysis. it is worth to develop it, because its biggest advantage is speed. This process should not take more time than the process of compiling the program, which makes it very attractive compared to the 22 hours mentioned in the literature (Park et al., 2009). As a result, it can be used as one of the functionalities of the IDE (e.g. real-time monitoring), because in a very short period of time the programmer will receive information about, for example, the phenomenon of order violation.

As mentioned in section 7, in order to reduce the amount of *false-positive error*, further research should be conducted into the relationships between operations. Another branch of research that can be conducted is the use of the criterion developed in this work, allowing to locate the phenomenon of the violation of order, together with the source code model of multithreaded applications to develop a method based on artificial neural networks.

REFERENCES

Abbaspour Asadollah, S., Sundmark, D., Eldh, S., & Hansson, H. (2017). Concurrency bugs in open source software: a case study. *Journal of Internet Services and Applications*, *8*, 4. https://doi.org/10.1186/s13174-017-0055-2

Abdulhamid, M., & Kinyua, N. (2020). Software for recognition of car number plate. *Applied Computer Science, 16*(1), 73–84. https://doi.org/10.23743/acs-2020-06

Andrew, J., Mcpherson, A. J., Nagarajan, V., Sarkar, S., & Cintra, M. (2015). Fence Placement for Legacy Data-Race-Free Programs via Synchronization Read Detection. *ACM Trans. Archit. Code Optim.*, *12*(4), 46. https://doi.org/10.1145/2835179

Bishop, M., & Dilger, M. (1996). Checking for Race Conditions in File Accesses. *Computing Systems*, 9(2), 131–152.

Cai, Y., Wu, S., & Chan, W. K. (2014). ConLock: a constraint-based approach to dynamic checking on deadlocks in multithreaded programs. In *Proceedings of the 36th International Conference on Software Engineering ICSE 2014* (pp. 491–502). https://doi.org/10.1145/2568225.2568312

Chen, D., Jiang, Y., Xu, C., Ma, C., & Lu, J. (2018). Testing multithreaded programs via thread speed control. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (*ESEC/FSE 2018*) (pp. 15–25). https://doi.org/10.1145/3236024.3236077

Chew, L., & Lie, D. (2010). Kivati: fast detection and prevention of atomicity violations. In *Proceedings of the 5th European conference on Computer systems* (*EuroSys '10*) (pp. 307–320). Association for Computing Machinery. https://doi.org/10.1145/1755913.1755945

Cygan, S., Borowik, B., & Borowik, B. (2018). Street lights intelligent system, based on the Internet of Things koncept. *Applied Computer Science, 14*(1), 5–15. https://doi.org/10.23743/acs-2018-01

Giebas, D., & Wojszczyk, R. (2018). Graphical representations of multithreaded applications. *Applied Computer Science*, *14*(2), 20–37. https://doi.org/10.23743/acs-2018-10

Giebas, D., & Wojszczyk, R. (2020a). Multithreaded Application Model. *Advances in Intelligent Systems and Computing*, *1004*, 93–103. https://doi.org/10.1007/978-3-030-23946-6_11

Giebas, D., & Wojszczyk, R. (2020b). Atomicity Violation in Multithreaded Applications and Its Detection in Static Code Analysis Process. *Applied Sciences*, *10*(22), 8005. https://doi.org/10.3390/app10228005

Giebas, D., & Wojszczyk, R. (2020c). Deadlocks Detection in Multithreaded Applications Based on Source Code Analysis. *Applied Sciences*, *10*(2), 532. https://doi.org/10.3390/app10020532

Jin, G., Song, L., Zhang, W., Lu, S., & Liblit, B. (2011). Automated atomicity-violation fixing. In *Proceedings of the 32nd ACM SIGPLAN Conference on Programming Language Design and Implementation* (*PLDI '11*) (pp. 389–400). https://doi.org/10.1145/1993498.1993544

Lu, S., Park, S., Seo, E., & Zhou, Y. (2008). Learning from mistakes: a comprehensive study on real world concurrency bug characteristics. In *Proceedings of the 13th international conference on Architectural support for programming languages and operating systems* (*ASPLOS XIII*) (pp. 329–339). https://doi.org/10.1145/1346281.1346323

Netzer, R., & Miller, B. P. (1992). What are race conditions? Some issues and formalizations. *ACM Letters on Programming Languages and Systems (LOPLAS)*, *1*(1), 74–88. https://doi.org/10.1145/130616.130623

Park, S., Vuduc, R. W., & Harrold, M. J. (2010). Falcon: fault localization in concurrent programs. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1* (pp. 245–254). https://doi.org/10.1145/1806799.1806838

Park, S., Zhou, Y., Xiong, W., Yin, Z., Kaushik, R., Lee, K. H., & Lu, S. (2009). PRES: probabilistic replay with execution sketching on multiprocessors. In *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles* (*SOSP '09*) (pp. 177–192). https://doi.org/10.1145/1629575.1629593

Savage, S., Burrows, M., Nelson, G., Sobalvarro, P., & Anderson, T. (1997). Eraser: a dynamic data race detector for multithreaded programs. *ACM Trans. Comput. Syst.*, *15*(4), 391–411. https://doi.org/10.1145/265924.265927

Torres, L. C., Marr, S., Gonzalez, B. E., & Mössenböck, H. (2018). A Study of Concurrency Bugs and Advanced Development Support for Actor-based Programs. *Lecture Notes in Computer Science*, *10789*, 155-185. https://doi.org/10.1007/978-3-030-00302-9

Vinesh, N., Sethumadhavan, M. (2020). ConFuzz—A Concurrency Fuzzer. *Advances in Intelligent Systems and Computing*, *1045*, 667-691. https://doi.org/10.1007/978-981-15-0029-9_53

Yu, Z., Zuo, Y., & Xiong, W. C. (2019). Concurrency Bug Avoiding Based on Optimized Software Transactional Memory. *Scientific Programming*, *2019*, 9404323. https://doi.org/10.1155/2019/9404323.