SOFTWARE NETWORKS IN THE LOGICAL ARCHITECTURE OF THE CYBER-PHYSICAL TRAFFIC SYSTEM

Milorad K. Banjanin[1], PhD; Mirko Stojčić[2], PhD; Dejan Drajić[3], PhD

[1] University of Novi Sad, Faculty of Technical Sciences, Trg Dositeja Obradovića 6, 21000, Novi Sad, Serbia, banjanin@uns.ac.rs

[2] University of East Sarajevo, Faculty of Transport and Traffic Engineering, Vojvode Mišića 52, 74000, Doboj, Bosnia and Herzegovina, mirko.stojcic@sf.ues.rs.ba

[3] University of Belgrade, Faculty of Electrical Engineering, Bulevar Kralja Aleksandra 73, 11120, Beograd, Serbia, ddrajic@etf.bg.ac.rs

Abstract: *Traffic and transportation are no longer exclusively tied to and limited by the physical world. The virtual or cyber world is becoming an integral part of almost all areas of human life and activity, including systems and processes in traffic and transport. The integration of the physical and virtual world is represented by the cyber-physical system (CPS), and the paper presents a model of logical architecture of such a system with a double learning loop (CPS^2) for the observed case study - section of the Motorway 9th January (M9J), a key road in Republika Srpska. Within the proposed model, the focus of the research is placed on software networks, which, among others, form the core of the virtual segment of CPS. A software network has been created that has the function of visualizing the way of interconnection of several applications, data processing software, embedded systems software, and network monitoring software and tools within the CPS model.*

Keywords: *CPS, CPS^2, model of logical architecture CPS, M9J, software networks*

1. INTRODUCTION

Today, traffic and transportation are no longer exclusively tied to and limited by the physical world. With the achieved development of information and communication technologies, the virtual or cyber world becomes an integral part of almost all areas of human life and activity, including systems and processes in traffic and transport. The integration of the physical and virtual world is represented by cyber-physical systems (CPS). They perform their significant role by acquiring data on physical variables in the vicinity of roads using sensors, traffic management using monitoring and control centers, automatic control of system elements (light signaling, in-vehicle systems), fast provision of information for users in various roles of the human factor in traffic, etc. The virtual or cyber component of CPS is based on the functions of embedded computers that monitor and control physical processes in a network in which many things happen in parallel, usually with feedback loops. This means that there are mutual influences between physical parallel processes and computer processes, which are defined by sequential steps. The feedback loop between physical and computer processes includes sensors, actuators, physical dynamics, calculations, software failures and network problems, communication delays. CPS is both a control system with automatically generated interfaces and a context-aware control system.

Thus, hardware network elements-sensors, microcontrollers and actuators, within CPS function integrally with cyber elements such as computer algorithms for data processing, for monitoring and control of physical processes, for decision making on storage locations and for data analysis. In this way, safety, mobility, efficiency, environmental protection and general optimization of traffic systems are significantly increased [1]. In addition to sensors, actuators and microcontrollers, autonomous units – „things“ are increasingly being networked and composed in the domain of the physical world, enabling instantaneous dynamic reconfiguration or adaptive programming. These elements make a resilient and adaptable IT ecosystem, which is a special type of system in which several systems with different degrees of autonomy achieve common goals while adapting to a given environment.

Within technological systems such as CPS, the existence of a large number of complex networks is evident. Each of them consist of a large number of nodes and links. They describe the interactions between the entities of which the network system is composed. In addition to nodes and links, the complexity of the network consists of various equipment, removable media, protocols, and hardware and software components of embedded computers. Various types of complex networks can represent integral parts of CPS, but the core of the virtual segment of cyber-physical systems consists of software networks. They belong to real complex networks with similar principles of ontological and researcher cooperation networks. Their characteristic is that they show: the small world effect (the length of the shortest path between two arbitrary nodes is small, drastically smaller in relation to the size of the network) and a high degree of local clustering (the nearest neighbors of the arbitrary node form a relatively dense subgraph). As a precursor to the inclusion of software networks in the logical architecture of CPS, the paper [2] can be emphasized, in which the cyber-physical system for traffic control is analyzed. In that paper, CPS is presented as an integration of information and transport processes. The approach to modeling CPS in this case is based on Cyberize the Physical (CTP), which means surrounding physical subsystems with software abstractions. The authors in [3] propose the creation of a cyber-physical system based on Software Defined Networking (SDN) and service-oriented architecture. A similar study is presented in [4] and aims to apply a software-defined approach to CPS creation to adequately address the most common CPS challenges, such as modeling, development, performance, process management, communication and coordination, scalability and error tolerance. Given that increasing congestion in traffic is evident in large cities, and thus parking problems, the research presented in [5] is focused on the development of CPS based on software agents for the smart parking system. Special focus is on the way agents are interconnected with physical controllers using appropriate Internet of Things (IoT) technologies. In [6], a cyber-physical system for managing city traffic signal is presented. The authors point out that the proposed CPS model eliminates the shortcomings of existing traffic signal control systems when it comes to management strategies, testing, simulation and practical application. The authors in [7] deal with the modeling and design of semantic, efficient, secure cyber-physical transport systems (CPTS), while in addition to applications in traffic systems, the paper [8] presents the possibilities of creating software defined CPS for industrial automation.

The subject and goal of the research presented in this paper is to create a logical architectural model of CPS in the real space of the Motorway 9th January (M9J), with an emphasis on the role and structure of the software network in its technical-technological and socio-humanistic complexity. The selected geo-area of the M9J, a key road in Republika Srpska, B&H, connects the cities of Banja Luka and Doboj for a length of 72 km. It is covered by the signal of the telecommunications network M:tel Banja Luka, as a communication platform for creating CPS.

The paper is structurally divided into five sections and a review of the used literature is given at the end. After the introduction, section 2 presents the results of research of current CPS models according to logical architecture. The created logical architectural model of CPS is presented in section 3, and in section 4 the software network is analyzed as a component of its structural and functional complexity in the interactions of physical and computer processes. Concluding remarks and directions for future research are listed in section 5.

2. CPS MODELING

CPS modeling is a key process for future technical development issues and it involves the application of knowledge from management engineering, software engineering, sensor networks, physics of dynamic states of complex systems and processes, etc. Some of the important characteristics of CPS, as an engineering system, are: it belongs to the cluster of artificial and not natural systems because it was created as a result of human creativity and for human needs; it is dimensioned from the aspect of technical complexity (complex flow of information, energy, mass, values) and humanistic-social complexity (significant influence of organizational factors); satisfies aspects of reality because it exists in the physical world; an open system with scalable capabilities and dynamic in time because it is not a fixed structure but a time-adaptable variable system or its components; operates in a hybrid combination of continuous states with discrete and mixed control mechanisms that is only partially human-intelligent with autonomous elements of intelligent subsystems.

Although the universal CPS reference model is not yet recognized in the world, the common CPS architecture contains models of physical processes, models of software and computer processes, computer platforms and networks. Current CPS models can be divided according to the architecture into [9]:

- Classic cyber-physical systems (CPS^0) that have the ability to control the feedback loop (decision making) in real time thus providing adaptability, reconfigurability, fast response and robustness.
- Cyber-physical systems in which physical and virtual components are integrated and which have an extended logical architecture in relation to CPS^0. This extension refers to the learning module (CPS^1). Such systems are called single learning loop systems, where learning can affect or change the physical part (settings, schedules, etc.).

- Cyber-physical systems with double learning loop (CPS^2) in which there is an additional virtual level with a learning module that has the ability to learn about the learning module of the CPS^1 model (learning about learning). Learning in this, but also in the CPS^1 model, is based on combining past, present and predicted data, researching advanced algorithms of artificial intelligence, fuzzy logic, multicriteria decision making, machine learning, deep learning, etc. [9].

The modeling in this paper is based on the concept of structuring holon systems, which means that CPS consists of several autonomous self-organizing systems such as complex networks that operate with the same goal, which is to provide telecommunications and digital services to identified and mobile users. Users are legally-regulatory participants in the implementation of certain classes of tele-traffic of a mobile service provider in real space and in real time, including a certain portfolio of digital services that are delivered through the IoT subsystem in the dynamic structure of the CPS.

3. LOGICAL ARCHITECTURAL MODEL OF CPS

The proposed CPS model with a double learning loop (CPS^2) for the observed case study - section M9J, is given in Figure 1. One layer of physical processes and two layers of digital processing can be clearly seen, which essentially form a double loop of such a model. Since CPS can be viewed as a network of physical and virtual elements in interaction, there are requirements for different technologies in their specific application, and network technologies are one of the most important [10]. CPS can use different network and communication models, but in this case the following types of networks are defined through which the proposed model operates:

1) M:tel provider network, especially cellular Long Term Evolution (LTE) technology. LTE is a fourth generation mobile network technology that covers the observed section of the motorway. It provides users with access to mobile telephony, but also other Internet services using an radio access network consisting of base stations [11];
2) Wireless sensor networks are part of the physical component of the CPS. They collect data on physical processes, ie. perform analog/digital conversion, thus enabling data to flow into the cyber component of the CPS;
3) Software networks. They represent networks of software entities with specific functions;
4) Transport networks. Transport networks can be represented by graphs that imply a set of nodes interconnected by branches, ie. links. Within the CPS, several interacting transport networks can be identified;
5) Traffic networks. The observed section of the motorway as a component of the CPS is part of the traffic network of roads of the Republic of Srpska and Bosnia and Herzegovina. In that network, nodes represent intersections and cities, and roads have the role of links that connect them. Specifically, the motorway section connects three junctions: Mahovljani, Prnjavor and Johovac.
6) Social networks. Mobile users today have access to a very large number of social websites, applications and platforms, with different functions and purposes. The concentration of users in the area of the motorway as a component of the CPS is increasing every day. This means that users in these locations, through the network of the mobile operator, can access various social networks.
7) Artificial neural networks. The adaptability property of CPS, among other things, is based on the application of machine learning techniques. Among them, artificial neural networks occupy a special position. The basis of virtual sensors, ie. learning modules and decision making modules located in the cyber component of CPS, are adaptive techniques based on artificial neural networks.

Each of these networks performs a specific function within the CPS, and their interaction builds an adaptive IT ecosystem. It is a special type of system in which multiple systems with varying degrees of autonomy achieve common goals by adapting to a particular environment in real time.

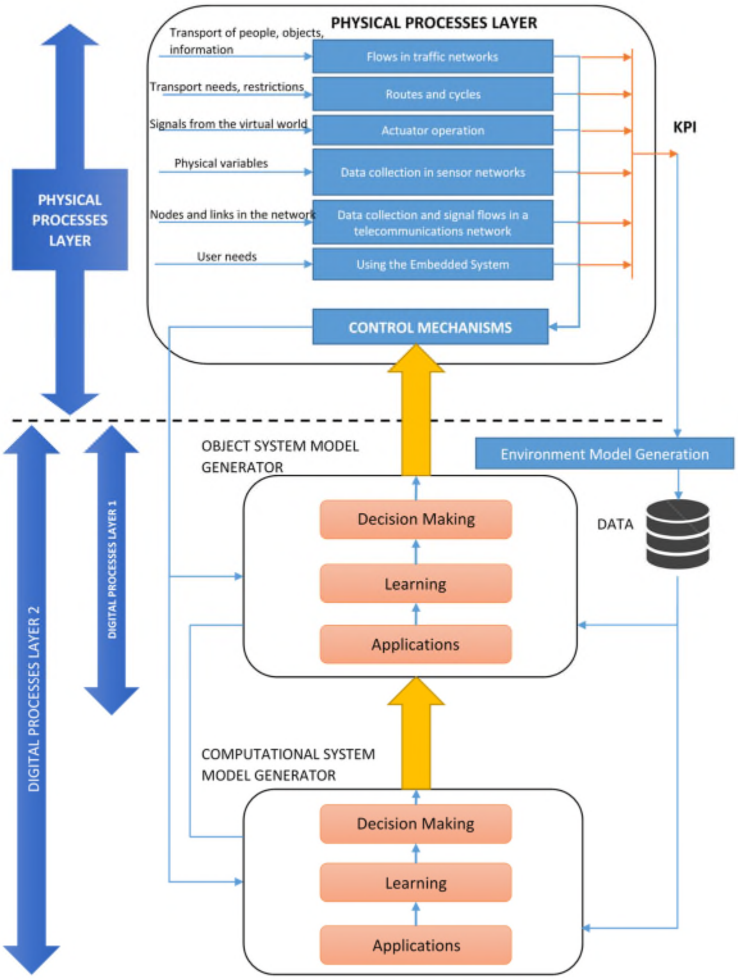
PHYSICAL PROCESSES LAYER
Transport of people, objects, information
Flows in traffic networks
Transport needs, restrictions
Routes and cycles
Signals from the virtual world
Actuator operation
KPI
Physical variables
Data collection in sensor networks
PHYSICAL PROCESSES LAYER
Nodes and links in the network
Data collection and signal flows in a telecommunications network
User needs
Using the Embedded System
CONTROL MECHANISMS
OBJECT SYSTEM MODEL GENERATOR
Environment Model Generation
DATA
DIGITAL PROCESSES LAYER 1
DIGITAL PROCESSES LAYER 2
Decision Making
Learning
Applications
COMPUTATIONAL SYSTEM MODEL GENERATOR
Decision Making
Learning
Applications

Figure 1. Logical architectural model of CPS

The CPS model in the area of the observed section of the motorway generates very large amounts of data (BigData) due to the existence of a large number of sensors. Therefore, for such systems, it is necessary to have resources for storage (Data Warehouse) and analysis of a large amount of data. The CPS model emphasizes the "Cloud" approach as a solution to this problem, ie. processing paradigm Cloud Computing. The central role in the cloud is played by the learning (and decision making) module-virtual sensors based on machine learning techniques. Virtual sensors base their work on aggregated data obtained from one or more physical sensors. Thus, they process the input data and give a new interpreted value. In addition, the task of virtual sensors is to send the processing results to a database-data warehouse, from where the Application Program Interface (API) can access them. The term Data Warehouse represents data extracted from operational databases and stored in special databases.

The following modules can be distinguished in the structure of the proposed CPS model:

- **Learning module and decision making module**. In the proposed CPS model, it is necessary to analyze and model a large amount of collected data in physical streams that flow into virtual sensors-network applications for processing and transmission to the output. The best solution lies in the field of adaptive models based on artificial intelligence that have the ability to learn. Therefore, the most common techniques and algorithms implemented by the learning module are based on machine learning. Decision making within the CPS is a process by which, based on several inputs and certain criteria, the best solution is selected between several alternatives. Solutions are found in mathematical models of multicriteria decision theory (MCDM) or in adaptive models. Virtual (software) sensors can perform learning and decision making functions based on various data obtained using physical sensors. The learning module and the decision making module are implemented on the SaaS (Software as a Service) layer of the Cloud.
- **IoT module.** In addition to sensors, actuators and microcontrollers in the domain of the physical world, autonomous things are increasingly being networked. They can communicate via the Internet, thus creating a smart environment [12]. IoT occupies a special place within the CPS, since the CPS is considered to be its advanced version. Networked objects have their physical and virtual representation in the digital world, are programmable and can be uniquely identified. Adaptability in IoT is reflected in the continuous adaptation of heterogeneous devices to dynamic changes in the environment. This means turning on, off, changing modes, all without human influence.

- **Energy management module**. The biggest challenge for designing engineering systems is the power supply of physical sensors, which is most often based on batteries. In addition to the batteries themselves, combinations with one of the technologies for collecting energy from the environment (Energy Harvesting) are possible. Since communication and sensor data processing consume most of the energy, consumption can be resolved to some extent in this domain. One of the possible solutions is the application of simpler signal processing algorithms, simpler modulation techniques and coding techniques. Sensor operating modes in which they sleep when inactive are a good energy saving strategy.
- **Traffic and people, things and information transport management module**. This module has the task of managing flows in traffic and transport networks within the CPS. Like other modules, it is based on adaptive models of machine learning. The inputs represent data on physical processes obtained with the help of physical sensors, and among others, the following can be singled out: flows in traffic networks, routes and cycles, transport needs and restrictions, etc. In this way, the module can solve a number of problems in the domain of transport networks. Examples include: routing, optimization problems, but also security threats in networks.

4. SOFTWARE NETWORK IN THE LOGICAL ARCHITECTURAL MODEL OF CPS

Figure 2 shows a software network in which several software systems (entities) are interconnected within the proposed CPS model. In this particular case, its function refers to the collection and processing of data related to the performance of the telecommunications network of the M:tel provider along the observed section, and the control and monitoring of the network and the embedded systems. In this case, the following are considered as software entities in the presented graph: mobile applications, data processing software, network monitoring software and tools, embedded systems software.

Mobile applications play a key role in the acquisition of data on classes of telecommunications traffic in the M:tel network, and the following are used:

- *Mp3 Music Downloader*. Allows you to search, play (stream) and download mp3 music formats. Some of the most important features are 1) Provides access to over a million high quality mp3 tracks; 2) One of the largest free mp3 music memories; 3) Search for music by title, artist, genre or album; 4) Quickly download mp3 high quality audio and play online or offline.
- *Video Downloader*. The free Video Downloader application provides search, playback and download of video content from the site tubidy.mobi. Key features: video playback using the built-in browser; video playback offline; all supported download formats (mp3, m4a, mp4, m4v, mov, avi, wmv, doc, xls, pdf, txt, etc.); download several files at once; continuation of failed downloads; supported download of HD videos, etc.
- *Network Cell Info Lite*. Provides cellular and Wi-Fi network monitoring. It is a tool for measurement/diagnostics based on logs (4G +, LTE, CDMA, WCDMA, GSM). The application can help solve reception problems and network connection problems, and at the same time provides information on the radio frequencies of the mobile operator and the parameters of the received signal. Includes an option to test WiFi/mobile Internet speed performance with a history of download, upload, ping and jitter results.
- *Google Map*. Over 220 countries and territories have been mapped and hundreds of millions of locations and places on the map have been marked. Uses GPS navigation to provide the user with real-time traffic information. Among others, the following characteristics are of particular importance: finding the best route with automatic recruitment based on traffic, accidents and closed roads, offline maps for searching and navigating without an Internet connection.

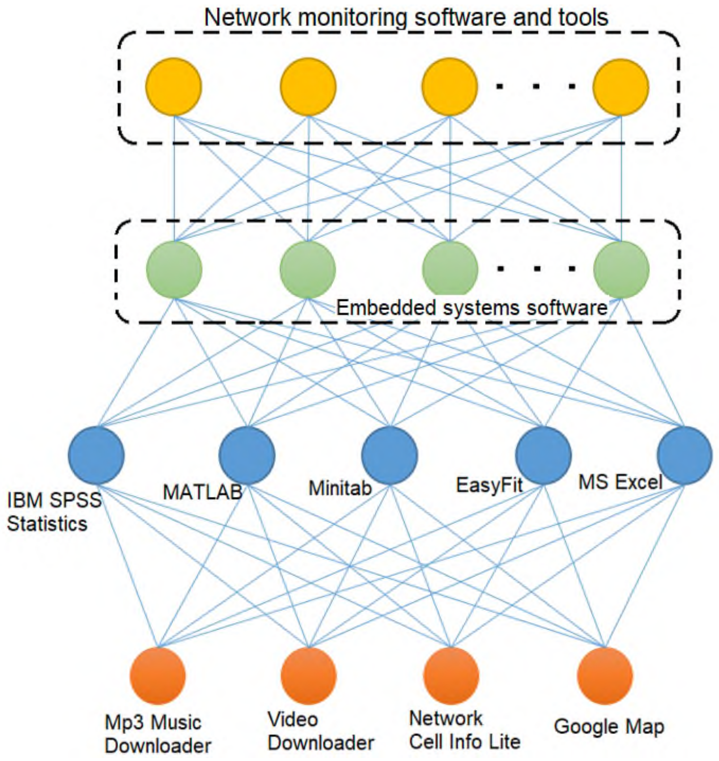

Figure 2. Software network model within CPS

The following software is used for data processing:

- *IBM SPSS Statistics* software platform provides a robust set of functions that allow useful data to be extracted from a large set of data. In addition to statistical modeling, it offers the possibility of creating predictive models based on the principles of machine learning, which is very useful in working with large databases (Big Data).
- *MATLAB* is a software package, a numerical computing environment developed by MathWorks. Allows matrix manipulation, function drawing and data visualization, algorithm implementation, user interface creation and connection to programs written in other languages.
- *Minitab* statistical software allows to spot trends, solve problems and view data. It is a comprehensive and one of the best tools in its class for data analysis and process improvement. Ensures that users get the most out of their analysis, and make better, faster and more accurate decisions.
- *EasyFit* allows you to automatically adjust the distribution to user data and select the best model in seconds. It is designed to make data analysis as easy as possible, leaving complex technical details in the background. Advantages of the software: reduces analysis time, prevents errors in analysis, easy to learn and use, etc.
- *MS Excel* software in this research acts as an intermediary between these applications and data processing software. Its function is data preprocessing in terms of organization, filtering, formatting, structuring data into vectors.

Network monitoring software tools can fall into two main categories: an agent-free platform and an agent-based platform. An agent-free platform is usually installed on a server or computer that is physically connected to the observed network. The advantage of this approach is that it does not require installation on each individual device, but also that it can automatically detect devices on the network. Agent-based platforms are implemented in the software as a service (SaaS) model, which means that they are accessed via the web. Agents or monitoring programs are installed on each individual device and in this way provide more detailed data compared to a system without agents. According to the site pcwdld.com, the 10 best network monitoring tools can be singled out, which can also be used within the proposed CPS model [13]:

- SolarWinds Network Performance Monitor - a leading network monitoring system that uses the Simple Network Management Protocol (SNMP) to check the status of network devices;
- Datadog Network Monitoring - allows good visibility of each of the network components and the connections between them;
- ManageEngine OpManager - SNMP-based and has automatic detection capabilities. It installs on Windows Server and Linux;
- Paessler PRTG - is a collection of tracking tools, running on a Windows server;
- Site24x7 Network Monitoring – cloud based network, server, and application monitoring, monitoring both physical and virtual resources;
- Nagios XI – infrastructure monitoring tool characterized by scalability with add-ons. Works on Linux;
- Zabbix – network, server and application monitoring tool. It is available for Linux, macOS and Unix;

- Incinga 2 – also an extensible tool that installs on Linux;
- Progress WhatsUp Gold - SNMP-based, extensible, and installed on Windows Server;
- ConnectWise Automate - includes many task automation tools and is installed on Windows Server [13].

The development of electronics, microprocessors and microcontrollers has led to the emergence of increasing use of embedded systems in modern cars, which require certain software to work. For example, communication protocols that allow the transmission of data and commands to the device in the vehicle cannot be controlled without a driver. In addition, software plays a key role in the proper reading and use of sensor-generated data. Today, cars have computer-controlled electronic systems, and the most common embedded systems in the vehicle include airbags, ABS, black box, adaptive cruise control, drive by vire, satellite radio, telematics, emissions control, traction control, automatic parking, in-vehicle entertainment systems, head up display, crash safety sensors, navigation systems, tire pressure monitoring, climate control, etc. [14].

Each entity in the software network is assigned a name that is used for referencing by other entities. There are two types of software networks depending on the type of entity and the interconnections, and they are:

- *homogeneous networks* - have entities of the same type, but also links that define the same relationships between nodes;
- *heterogeneous networks* - have entities of different types and/or links that represent different relationships between nodes.

Since the network shown in Figure 2 is composed of entities of different types, it is a heterogeneous software network. In this case, the links connect the entities at different levels and are called vertical. In contrast, all links for homogeneous software networks are horizontal. Two popular heterogeneous networks are:

- *Hierarchical tree* that contains all entities defined in software systems, and records vertical dependencies between entities;
- *General Dependency Network (GDN)* - types and names are defined for nodes, and types and weights (link strengths) for links. GDN defines the following types of relationships represented by links: call relationship between functions, reference relations between entities at the packet level, reference relations between entities at the class level, use relationship between functions and variables, relationship contains reflecting entity hierarchy.

Different types of software networks are determined by dependencies, interactions, relationships or cooperation between software entities that can be defined with three levels of abstraction: level functions, level classes, level packets. When it comes to collaboration, specific software networks can link packages, classes, and methods. Software entities have the property that their internal complexity is viewed as a function of the complexity of its components (number of components). Modern software systems are very complex, and a certain metric is used to assess complexity, and one of the most common is LOC (Lines of Code). In addition, cyclomatic complexity, Halstead metrics, etc. are often used which are mainly oriented towards the internal complexity of software entities. The large number of software entities, the interactions between them, and the states in which the system can be found, make the software complex [15]. The interactions between the entities themselves are also complex, and the level of complexity is determined by the metric software design classes. It is reflected by conjunctions, cohesion, inheritance, and representation.

One of the basic design principles in software engineering is "Low coupling, high cohesion". This means that, when it comes to the connection between the modules of the software system, it must be the least possible maintenance. At the same time, strong relationships must be maintained between the elements of each module. This principle highlights the encapsulation and modularity of software systems. External links are of great importance for assessing the cohesion of software modules. This means that a module cannot be considered to have strong cohesion if there are many more external dependencies. Thus, cohesion metric represents the strength of the connection of elements within a module and is based on two forms of coupling: data and call coupling. The most famous metric of this type is LCOM (Lack of Cohesion of Methods). Graph Clustering Evaluation (GCE) metrics can be applied to a graph representing software systems, in order to assess the degree of cohesion of software entities.

The constituent parts of software entities enable the expression of certain features of them. An example is the internal complexity of a high-level entity, which is a function of the internal complexity of its constituent parts. To calculate this entity composition metric, hierarchical trees are used. It should be noted that hierarchical trees can be used alone or in combination with other software networks.

Based on the analysis of previous relevant research, several properties are known by which software metrics satisfy the theoretical lack of cohesion metrics: *Non-negativity*. Cohesion metrics cannot take a negative value; *Normalization*. The metric value belongs to the interval [0, M], where M is a fixed maximum value; *NULL value*. A software entity has a cohesion value equal to NULL if the set of relationships within the software entities (Rc) is empty; *Maximum value*. The cohesion metric has a maximum value if Rc is maximal; *Monotonicity* is expressed through the following inequalities:

$$C(e) \leq C(e'), \tag{1}$$

$$L(e) \geq L(e'), \tag{2}$$

where *e* is the software entity; *e*' is the software entity such that $R_e \subseteq R_{e'}$; C denotes cohesion; L denotes a lack of cohesion metrics.

Connecting the property. The assumption is that e_1 and e_2 are two software entities between which there is no connection, and *e* represents the union of e_1 and e_2. Then the following inequalities apply:

$$C(e) \leq max\{C(e_1), C(e_2)\} \tag{3}$$
$$L(e) \geq min\{L(e_1), L(e_2)\}. \tag{4}$$

The architectural entities of a software system are represented by packages, classes, functions, global variables. The extraction of software networks takes over their identification and the identification of their interdependence. Programming languages, ie. programming languages of special tools usually extract only certain types of software networks. Nevertheless, SNEIPL, an extensible and independent language, is used today as an increasingly common approach to software network extraction. Its basic feature is to use the enriched Concrete Syntax Tree (eCST) representation of the source code to form software networks. In addition to eCST, other forms of tree for presenting source code are widely used, such as: Concrete Syntax Tree (CST) and Abstract Syntax Tree (AST). The basic architecture of SNEIPL consists of two components: a GDN extractor and a GDN filter that extracts the GDN form into a set of software network outputs.

Today's modern software systems consist of a multitude of interconnected entities at different levels of abstraction. Depending on the degree of abstraction, specific software networks differ, such as package, class, and method collaboration networks. Different types of entity connections of the same type determine different software networks. The importance of extraction and analysis of software networks is reflected in several areas: analysis of the complexity of the design and evolution of software systems; calculation of software metrics that reflect the quality of software system design; forming a fact base in the process of reverse engineering of software systems.

Software networks can be used for: Visualization of software systems [16]; Clustering of software entities in the process of identifying system architecture at a high level of abstraction [17]; Identification and removal of suspicious parts of the source code [18]; Locating concepts in source code [19]; Identification of design patterns; Error prediction in software systems [20].

5. CONCLUSION

The paper presents the logical architectural model of CPS in the geo-space of the M9J, when the focus is placed on the software network as one of its constituent components. A software network connects software entities in a virtual or cyber component of a CPS. A cyber-physical system can be considered an intelligent system, while the integration of different types of networks and devices makes CPS a heterogeneous environment of observed real processes in real time with scalable capabilities. This conclusion is derived from the analysis of the graph of the software network shown. For a specific problem, it is possible to add the appropriate software - nodes to the network and connect them with links to the rest of the network. In order to fully realize the benefits offered by CPS, various potential problems that may arise during its lifetime must be taken into account. One such problem is the interception and interference of communication between different components in the system. Therefore, it is necessary to carefully implement security mechanisms in the presented CPS model. A comparative analysis of previously published CPS research can conclude that the originality of this CPS model lies in the implementation of the software network in its logical architecture, which is of great practical importance because in the real world users can use a wide range of digital IT ecosystem services. At the same time, it should be added the fact that a key road in the Republic of Srpska M9J was used in the modeling case study, which is a special dimension of social utility with innovative economic implications of CPS. Future directions of research can be oriented towards the expansion of the software network with additional applications and software for data processing, but also the spatial expansion of the model beyond the observed section of the motorway, which will increase the functionality of the software network and CPS.

REFERENCES

[1] Edited by: Lipika Deka and Mashrur Chowdhury. Transportation Cyber-Physical Systems. ISBN 978-0-12-814295-0, Elsevier, 2019.

[2] Jianjun, S., Xu, W., Jizhen, G., & Yangzhou, C. The analysis of traffic control cyber-physical systems. Procedia-Social and Behavioral Sciences, 96, 2013: 2487-2496. https://doi.org/10.1016/j.sbspro.2013.08.278.

[3] Yu, H., Qi, H., & Li, K. A powerful software-defined cyber-physical system to expand CPS adoption. Software: Practice and Experience. 2019. https://doi.org/10.1002/spe.2728.

[4] Kathiravelu, P., Van Roy, P., & Veiga, L. SD-CPS: software-defined cyber-physical systems. Taming the challenges of CPS with workflows at the edge. Cluster Computing, 22(3), 2019: 661-677. https://doi.org/10.1007/s10586-018-2874-8.

[5] Sakurada, L., Barbosa, J., Leitão, P., Alves, G., Borges, A. P., & Botelho, P. Development of agent-based cps for smart parking systems. In IECON 2019-45th Annual Conference of the IEEE Industrial Electronics Society; 2019: Vol. 1, pp. 2964-2969. IEEE. 10.1109/IECON.2019.8926653.

[6] Zhang, L. L., Zhao, Q., Wang, L., & Zhang, L. Y. Research on Urban Traffic Signal Control Systems Based on Cyber Physical Systems. Journal of Advanced Transportation, 2020. https://doi.org/10.1155/2020/8894812.

[7] Petnga, L., & Austin, M. A. Tubes and Metrics for Solving the Dilemma-Zone Problem. In The Tenth International Conference on Systems (ICONS 2015), Barcelona, Spain; 2015:119-124.

[8] Ahmed, K., Blech, J. O., Gregory, M. A., & Schmidt, H. Software defined networking for communication and control of cyber-physical systems. In 2015 IEEE 21st International Conference on Parallel and Distributed Systems (ICPADS); 2015: 803-808. IEEE. 10.1109/ICPADS.2015.107.

[9] Putnik, G. D., Ferreira, L., Lopes, N., & Putnik, Z. What is a Cyber-Physical System: Definitions and models spectrum. Fme Transactions, 47(4), 2019: 663-674. 10.5937/fmet1904663P.

[10] Jawhar, I., Al-Jaroodi, J., Noura, H., & Mohamed, N. Networking and Communication in Cyber Physical Systems. In 2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW); 2017: 75-82. IEEE. 10.1109/ICDCSW.2017.31.

[11] Banjanin M.K., Stojčić M, Drajić D, Ćurguz Z, Milanović Z, Stjepanović A. Adaptive Modeling of Prediction of Telecommunications Network Throughput Performances in the Domain of Motorway Coverage. Applied Sciences. 11(8), 2021, 3559. https://doi.org/10.3390/app11083559.

[12] Drajić D. UVOD U IoT (Internet of Things). Akademska misao Univerzitet u Beogradu – Elektrotehnički fakultet Beograd, 2017.

[13] https://www.pcwdld.com/best-network-monitoring-tools-and-software (Accessed: 28.05.2021)

[14] https://www.watelectronics.com/importance-of-embedded-systems-in-automobiles-with-applications/ (Accessed: 28.05.2021).

[15] Banjanin, K.M. Komunikacioni Inženjering; Univerzitet u Istočnom Sarajevu, Saobraćajno-Tehnički Fakultet Doboj: Doboj, Bosnia and Herzegovina, 2007; ISBN 978-99938-859-4-8.

[16] Lanza, M., and Ducasse, S. Polymetric views—a lightweight visual approach to reverse engineering. IEEE Transactions on Software Engineering 29, 9 (Sept.); 2003: 782–795. 10.1109/TSE.2003.1232284.

[17] Scanniello G, D’Amico A, D’Amico C, D’Amico T. Using the Kleinberg algorithm and Vector Space Model for software system clustering. In: Proceedings of international conference on program comprehension. IEEE Computer Society, 2010: 180–189. 10.1109/ICPC.2010.17.

[18] Oliveto, R., Gethers, M., Bavota, G., Poshyvanyk, D., & De Lucia, A. Identifying method friendships to remove the feature envy bad smell: NIER track. In 2011 33rd International Conference on Software Engineering (ICSE); 2011: 820-823). IEEE. 10.1145/1985793.1985913.

[19] Scanniello, G., & Marcus, A. Clustering support for static concept location in source code. In 2011 IEEE 19th International Conference on Program Comprehension; 2011: 1-10. Ieee. 10.1109/ICPC.2011.13.

[20] Bhattacharya, P., Neamtiu, I., & Shelton, C. R. Automated, highly-accurate, bug assignment using machine learning and tossing graphs. Journal of Systems and Software, 85(10), 2012: 2275-2292 10.1016/j.jss.2012.04.053.