TRENDS AND ACTUAL TECHNICAL SOLUTIONS FOR IMPLEMENTATION OF DIGITAL ECOSYSTEMS

Alexandru Averian[245]
DOI: https://doi.org/10.31410/EMAN.2018.620

Abstract: *In competitive markets a successful business strategy requires development of new innovative services and products to acquire a higher value for the client. Developing the new products is realized by collaboration of organizations that contribute to the added value of the service or product. The term of business ecosystem is used to better describe organizational collaboration that produces services of better value to the customer. Digital business ecosystems are counterparts of business ecosystems exploiting self-organizing properties of biological ecosystems which could include context-aware, self-organizing and scalable architectures that can be implied in solving complex problems. The present article evaluates existing technologies and protocols and proposes a set of solutions for implementing environment layer from the architecture of species that populate a digital ecosystem. We analyze the new service oriented middleware platform proposed for the modern auto vehicles, and evaluate the possibility to adopt this technology into digital ecosystems.*

Key words: *digital ecosystems, environment, context, digital species*

1. INTRODUCTION

Digital ecosystems are a new type of application that relies on a "universal digital environment" populated by digital entities which forms digital communities that evolve and interact by information exchange and who trade digital objects produced in or through the system [1]. Entities that participate and form the ecosystem can be applications running on simple wearable devices or services executed on smartphones, personal computers, or servers in data centers.

A digital ecosystem is an open system, any entity can connect to the system (at least theoretically) and interact with other entities to achieve the proposed objectives. The interconnection of heterogeneous devices can easily be achieved in a laboratory environment, but in reality it is difficult to obtain. To be able to communicate and interact in a dynamic and open system, a standardization is needed at the level of communication protocols, services, interfaces, and service semantics. Interoperability in a heterogeneous environment through common interfaces can provide new entrants with the ability to collaborate or compete with those already connected without making them dependent (or blocked) on a proprietary technology. This paper analyze existing technologies and protocols and proposes a set of technical solutions for implementation of environmental and context layers found in the architecture of digital objects that populate a digital ecosystem as described in [2].

This paper is structured as follows. Section 2 presents the candidate systems for use in the implementation of the communication medium for digital ecosystems. Section 3 presents SOME/IP - a new service oriented middleware. Last section presents the conclusions and hints for future work.

[245] Politehnica University of Bucharest, Splaiul Independentei nr. 313, Sector 6, Bucharest, Romania

2. IMPLEMENTING THE DIGITAL ENVIRONMENT

ALEXANDRU AVERIAN

Lecturer drd, Bucharest, Romania
Born: 1976
Interests: digital ecosystems, parallel and distributed computing, advanced information technologies and programming.

Details:
A. Averian received his B.S. in computer science (2002) from University of Bucharest, Romania, in present he is studying digital ecosystems as PhD student at Politehnica University of Bucharest. Starting with 2002, he has been lecturer of computer science at Spiru Haret University. His current research interests include different aspects of social and natural computing, digital ecosystems, modelling and simulation, software reliability and agile methodologies.

The environment level represents the way of communication between digital objects that inhabit a digital ecosystem. Digital environment assures communication between digital objects and allows for the exchange of data, messages and commands between entities living within the ecosystem. We cannot use a centralized, client-server model because it is not resilient, is not error tolerant, is not scalable and can be vulnerable to attacks. The environment will generally be a peer to peer (P2P) system because it has a number of advantages over the centralized model. These advantages result from the network definition, a P2P system is defined as a network in which all nodes are equivalent to each other, in the sense that all nodes can execute (in principle) the same set of functions required for network operation. P2P is defined at the application level, it offers the possibility of sharing data and computing resources (files, digital objects, processing services), being overlapped over the physical infrastructure of the Internet. The most important features of a P2P network are:

- resource sharing through direct transfer without intermediaries or centralized servers, in some cases centralized servers can be used for network initialization, node management;
- no central nodes, no central failure points, without central attack points;
- nodes actively participate in operations such as information management, resource search, data storage and management;
- the network has the ability to adapt to its variations in connectivity, to topology changes, the ability to reconfigure itself for an error;
- P2P network topology is tolerant to defects, having self-organizing ability to maintain network functionality and performance;
- the network can be structured or not, the physical proximity between the nodes is not important;
- the connections between the nodes are TCP connections, but can be represented by pointers to services;
- maintaining the network and verifying the connectivity is done by sending periodically some *ping* messages, in the case of detecting the fall of some nodes, the site can establish new connections.

In the following figure we have a classification of machine to machine (M2M) communication systems that can be used to implement the environment level of digital ecosystems. These can be divided into four major categories: Remote Procedure Call (RPC), Object/Component Oriented Middleware (OCOM), Transaction Oriented Middleware (TOM), and Message Oriented Middleware (MOM).

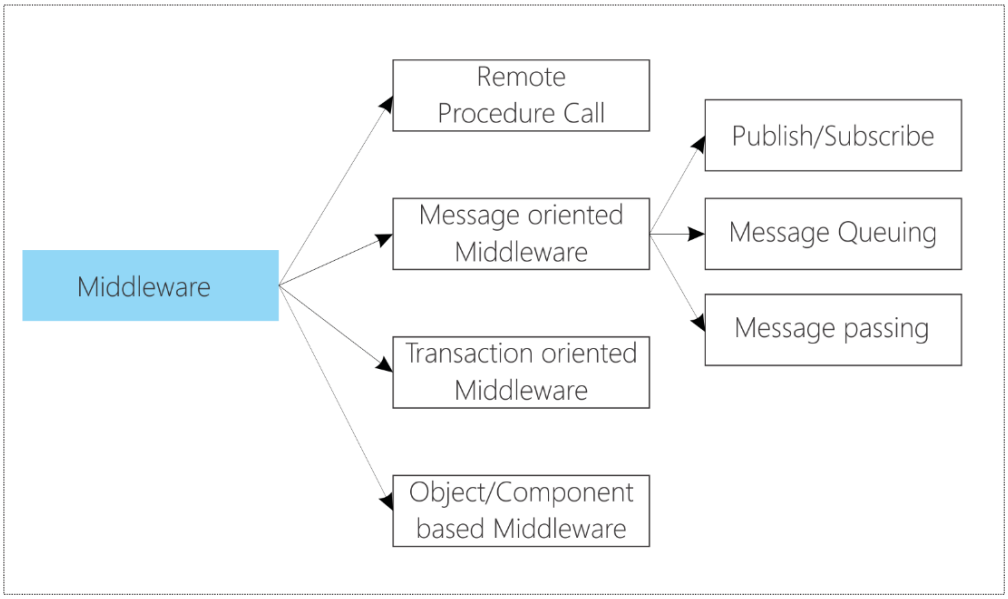

Figure 1. Classification of M2M communication systems

2.1 Remote procedure call

A Remote Procedure Call (RPC) based system offers an infrastructure that invokes the procedures that will run on remote servers. It typically comes in the form of an API that allows remote synchronous calls and abstracts the communication details. This model is not scalable, not error-tolerant, is best suited for distributed applications based on a client-server communication model. The following figure describes the architecture of applications using remote procedure call.

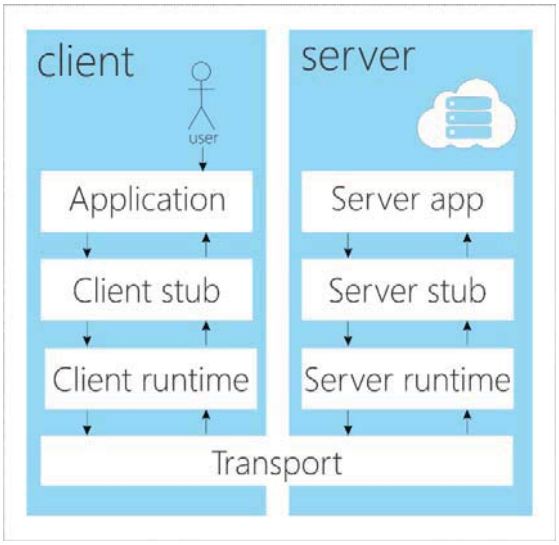

Figure 2. Remote procedure call

2.2. Object/Component based middleware

A Component based middleware is using an object-oriented programming model that allows communication between remote objects. This type of system extends the RPC model by adding new concepts specific to object oriented programming languages such as: encapsulation, inheritance, exceptions, etc. The system allows for (a)synchronous invocation of methods of interfaces that represents remote services. Such a system has limited scalability and is not suitable to implement the environment level for digital ecosystems.

2.3. Transaction Oriented Middleware

A Transaction Oriented Middleware is primarily used in database applications that perform different transactions in a distributed environment. The system ensures the accuracy of transactions between heterogeneous hosts, but introduces a significant overhead in operation. It allows both synchronous and asynchronous operation and is suitable in the interaction between application servers and database management systems. The QoS guarantees provided by this type of middleware are not always necessary.

2.4. Message Oriented Middleware

Message Oriented Middleware (MOM) is a family of middleware products that facilitates messaging across distributed systems. MOM systems offer a number of features such as synchronous and asynchronous communication, data transformation capabilities, application decoupling (or weak coupling), parallel message processing and multiple QoS levels. A MOM system uses one of the following communication paradigms: message passing, indirect queuing, publish/subscribe communication (data is published through a topic, and customers receive all messages posted by the topic they are subscribed to). In this category of systems we will analyze two products that support the publish/subscribe model, namely Message Queue Telemetry Transport and Data Distribution System.

Message Queue Telemetry Transport (MQTT) is a messaging protocol introduced by IBM in 1999, standardized by OASIS in 2013 [2]. MQTT is optimized for centralized data collection and analysis, connecting sensors, mobile and embeded devices to data processing applications running in data center. Sensors, devices and applications communicate through a message broker running on a server. All operations use a routing mechanism (one-to-one, one-to-many, many-to-many) which enable it as on optimal protocol for IoT. MQTT is composed of three components, publisher, subscriber and broker. Any device interested in some subject will register as a subscriber for specific topics and the broker will inform it when publishers emit data in interested topics. The following figure describes the architecture of the MQTT system.

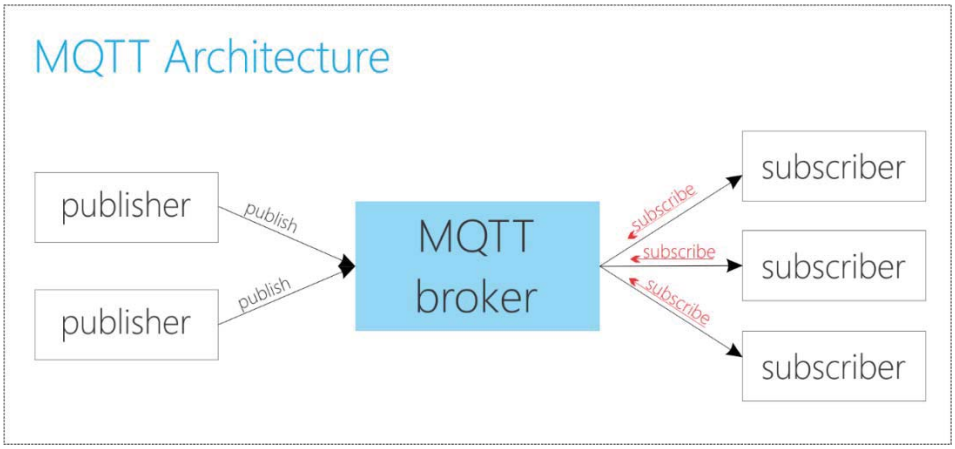

Figure 3. The architecture of the MQTT system

Data Distribution System (DDS) is a publish-subscribe protocol for real-time M2M communicatios developed by OMG [3]. DDS is a good choice for distributed processing of data coming directly from sensors, devices and applications without using any centralized IT infrastructure. The following figure shows the components that are contained in a M2M publish/subscribe communications framework such as DDS [4].

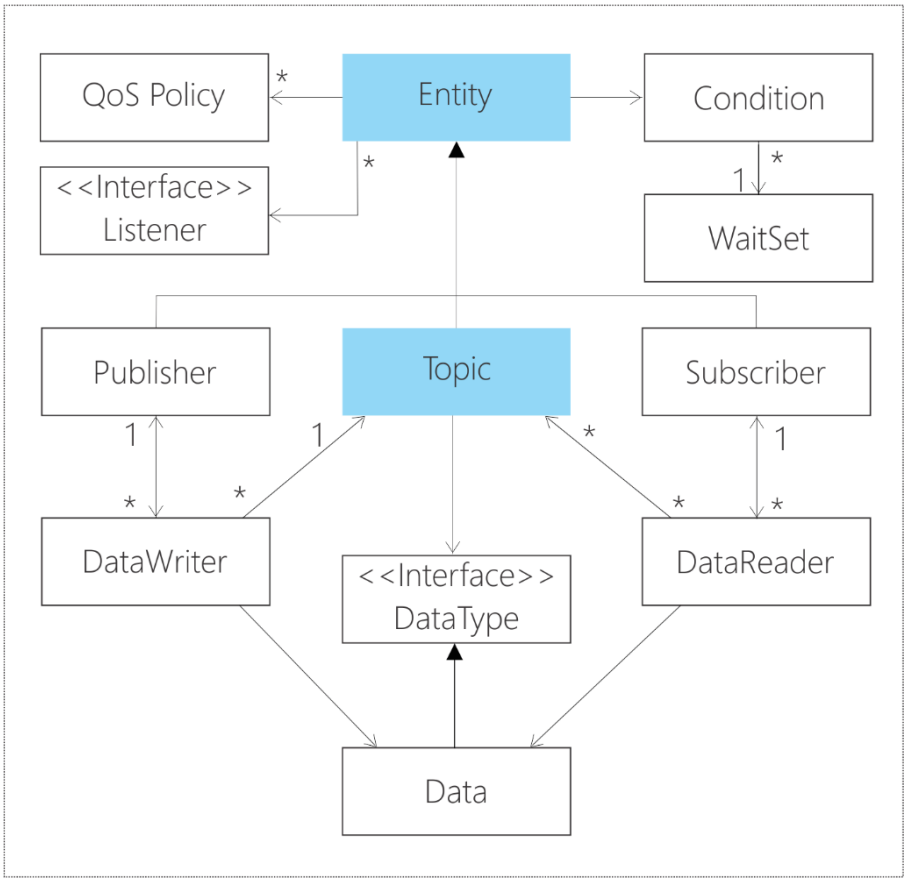

Figure 4. The components of a DDS middleware

A M2M middleware with publish/subscribe communication scheme exposes the following features:

- no central failure point - each application has a clear picture of what happens in the system, but it only depends on the data from its own context. Application failure affects only applications that depend on it.
- redundancy - multiple publishers may publish the same type of data, if one of the emitters is out of the system, a subscriber will receive further data from the emitters remaining in the system;
- discovery protocol allows applications to automatically recognize and exchange data based on the used topic.

Of the three communication models mentioned, publish/subscribe is best suited for building the environment level as it ensures asynchronous, scalable, many-to-many communication. In this scheme, the messaging emitters communicate with the subscribers, without prior knowledge, through a distributed P2P infrastructure. The system supports data filtering and allows a decoupling in terms of time, space and synchronization.

Time-based decoupling permits the broadcaster and receiver to cooperate directly and to communicate without having to be online at the same time. Decoupling in space refers to the fact that the transmitter and receiver are unaware of each other, their identity and location is not relevant. Synchronization decoupling refers to the fact that the receivers and transmitters do not have to synchronize, communication is accomplished by asynchronous notifications implemented with a callback mechanism.

Data filtering at reception can be implemented in two ways: subject-based filtering and content-based filtering. Subject-based filtering requires that messages are specifically labeled as belonging to a topic. In this way, the receiver will only receive messages that belong to a relevant topic and to which he has subscribed. Content-based filtering allows the delivery of messages to a handset only if the content of the messages matches a criteria set by the receiver. Publish/subscribe systems offer a range of QoS policies that provide a variety of properties such as: volatility, persistence, sustainability, prioritization and delivery.

For sending messages to the environment or for exchanging data between entities, a RPC communication scheme will be used. In 2017 OMG issued a standard for implementing a RPC protocol over DDS. Although the first implementations appeared from 2014, the standard was published in April 2017. It can be downloaded at http://www.omg.org/spec/DDS-RPC/1.0/. The general scheme of the RPC protocol over DDS can be seen in the following figure.

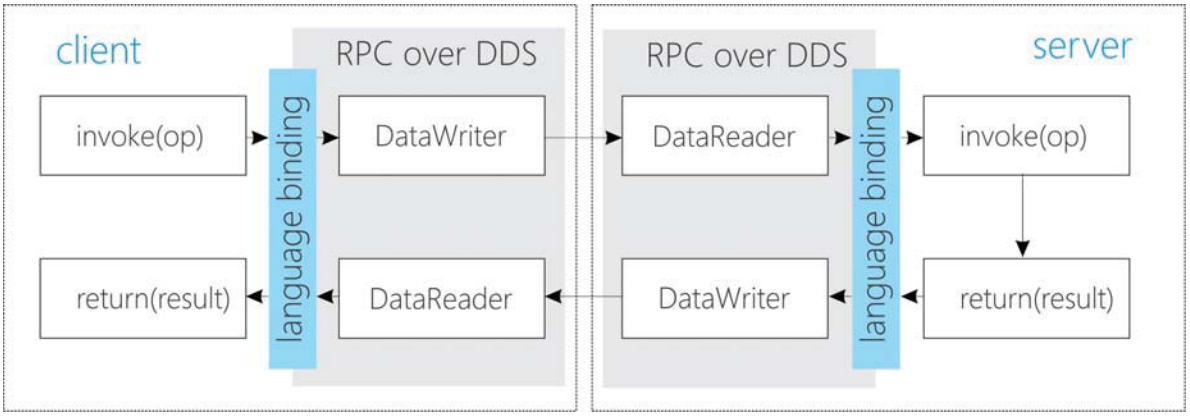

Figure 5. Implementation of RPC scheme over publish/subscribe

Depending on the type of application, the environment level can be implemented through a real-time data distribution system (DDS) or other messaging-oriented middleware such as: Extensible Messaging and Presence Protocol (XMPP), Advanced Message Queuing Protocol (AMQP), Message Queue Telemetry Transport (MQTT), or Constrained Application Protocol (COAP). These systems are used to realize machine-to-machine interconnections in IOT applications [5], [6].

3. SCALABLE SERVICE-ORIENTED MIDDLEWARE OVER IP

Automotive technology has evolved over time, turning cars into from a simple internal combustion engine with wheels to moving fusion of integrated computer systems. The evolution continues by integrating vehicles into an extended ecosystem in which they will communicate with each other and with the road infrastructure they use. The following are examples of technologies which are being developed and deployed that relate to automotive ecosystem:

- connected cars through mobile or Wi-Fi, these systems will allow not only functions such as real time traffic information, and video streaming but also remote diagnostics and updating of firmware;
- vehicle to vehicle communications will be used for cars to coordinate with each other in a digital ecosystem;
- augmented reality dashboards will provide information about objects on the road, if the driver looks towards an object the car will zoom in to see far away objects;
- intelligent charging of electric and hybrid vehicles;

- vehicle remote access, check vehicle fill level via smartphone app;
- autonomous vehicles will be able to drive passengers and make deliveries without any driver.

As part of this evolution SOME/IP is introduced as a communication middleware which interconnect and extend other in car communication protocols like CAN, LIN and FlexRay [7]. SOME/IP is the only known middleware that was designed to be integrated into AUTOSAR 4.x release [8]. While other middleware solutions often only support single communication features (RPC or publish/subscribe), SOME/IP supports a wide range of middleware features [9], as one can see in the next figure:

- serialization – transforming into and from on-wire representation;
- remote procedure call – implementing remote invocation of functions;
- service discovery– dynamically finding the functionality and configuring its access;
- publish/subscribe – dynamically configuring which data is needed and shall be sent to the client;
- fire and forget, events and event groups, field modification;

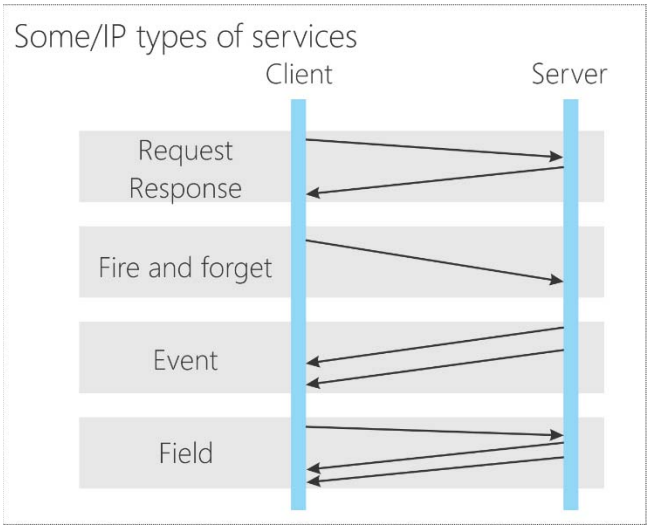

Figure 6. SOME/IP communication models

Some possible disadvantages of SOME/IP [10] are: computational overhead due to complex architecture, less suitable for hard real time systems and safety critical system (yet good for infotainment and driver assistance system with high requirements on data rate).

4. ANALYSIS AND CONCLUSIONS

This paper assesses existing technologies and protocols and proposes a set of solutions for implementation of environment and context layers present in the infrastructure of digital ecosystems. This work follows the study conducted in [11] and continue the research that was presented in respective paper. In the Message Oriented Middleware category we evaluated two products that support the publish/subscribe model, namely Message Queue Telemetry Transport and Data Distribution System. From automotive realm we introduced and evaluated SOME/IP middleware, conclusions can be observed in the following table.

	Data-centric middleware DDS	Message oriented middleware MQTT	Service – oriented middleware SOME/IP
Architecture	decentralized, peer to peer	centralized, server-based	bus, peer to peer
Administration	often autonomous	it involved configuration	minimal initial configuration
Discovery	dynamic discovery service	no, clients configured with server address	discovery mechanism
Transport	udp, tcp	tcp	udp, tcp
Security	tls	tls	no
Rpc	emulated	no	yes
Publish/subscribe	yes	yes	yes
Events	by data	by message	yes, event groups
QoS	yes	minimal	intrinsec
Content awareness	data-centric, content filtering and routing	data-agnostic	data-agnostic
Real-time	must keep up with data	as responsive as possible	as responsive as possible
Usage	manufacturing control, monitoring of power grid, control of robotics and unmanned vehicles, automotive driver safety and infotainment, aerospace and defense control, high-performance computing, low latency finance, real-time trading	mobile device to server messaging, point of sale, patient monitoring, remote resource monitoring, automotive telematics	automotive driver safety and infotainment, could be extended to vehicle remote access
In time evolution	minimize future complexity and deployment time, evolution	require administration and integration efforts	could be extended to accommodate vehicle to vehicle and vehicle to infrastructure communication

From digital ecosystem perspective a communication middleware with support of more communication schemes like RPC, pub/sub and events is more suited for building the environment level in digital ecosystems. The systems with pub/sub communication scheme where used in development of context models for digital ecosystems, as described in [12] article. Future studies will focus on expanding and automating the use of some/ip along with common-api technology from GENIVI [13]

ACKNOWLEDGEMENTS

The work has been funded by the Sectoral Operational Programme Human Resources Development 2007-2013 of the Ministry of European Funds through the Financial Agreement POSDRU/159/1.5/S/132395.

REFERENCES

[1] P. Dini *et al.*, "Beyond interoperability to digital ecosystems: regional innovation and socio-economic development led by SMEs," 2008.

[2] D. Locke, "MQTT V3.1 Protocol Specification," *IBM*, 2013. [Online]. Available: http://public.dhe.ibm.com/software/dw/webservices/ws-mqtt/mqtt-v3r1.html.

[3] "Data Distribution Service for Real-time Systems," *OMG*, 2015. [Online]. Available: http://www.omg.org/spec/DDS/1.4/.

[4] J. M. Cruz, A. Romero-garcés, J. Pedro, B. Rubio, R. M. Robles, and A. B. Rubio, "A DDS-based middleware for quality-of-service and high-performance networked robotics," 2012.

[5] E. Borgia, "The Internet of Things vision : Key features , applications and open issues," *Comput. Commun.*, vol. 54, pp. 1–31, 2014.

[6] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Commun. Surv. Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[7] A. Kern, "Ethernet and IP for Automotive E / E-Architectures Technology Analysis , Migration Concepts and Infrastructure," 2012.

[8] M. Weber, "The Future of Ethernet in AUTOSAR," in *Autosar Open Conference*, 2014.

[9] J. R. Seyler, T. Streichert, M. Glaß, N. Navet, and J. Teich, "Formal analysis of the startup delay of SOME/IP service discovery," in *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2015, pp. 49–54.

[10] J. Diemer, D. Thiele, and R. Ernst, "Formal worst-case timing analysis of Ethernet topologies with strict-priority and AVB switching," in *7th IEEE International Symposium on Industrial Embedded Systems (SIES'12)*, 2012, pp. 1–10.

[11] A. Averian, "Supply chain modelling as digital ecosystem," in *Proceedings of International Scientific Conference ITEMA 2017*, 2017, pp. 27–35.

[12] A. Averian, "Towards More Context-Awareness in Reactive Digital Ecosystems," in *Creativity in Intelligent Technologies and Data Science: Second Conference, CIT{&}DS 2017, Volgograd, Russia, September 12-14, 2017, Proceedings*, A. Kravets, M. Shcherbakov, M. Kultsova, and P. Groumpos, Eds. Cham: Springer International Publishing, 2017, pp. 640–654.

[13] GENIVI, "CommonAPI SOME/IP C++ User Guide," *GENIVI*, 2013. [Online]. Available: https://docs.projects.genivi.org/ipc.common-api.cpp-someip-tools/3.1.3/pdf/CommonAPISomeIPCppUserGuide.pdf.