Industrial service oriented architecture: Application to a milk processing plant

Itu Alina[1,2]
[1]Transilvania University of Brasov, Department of Automatics, Brasov, Romania
[2]Corporate Technology, Siemens SRL, Brasov, Romania

***Abstract*— We introduce a workflow for optimizing a milk processing application using a service-oriented architecture containing three layers. The main objectives of the development were: a minimized processing time, automated usage of the optimized processing plan, as well as application flexibility. The optimization problem is formulated as mixed-integer programming (MIP) application, and the resulting model is solved using a linear constraint satisfaction problem solver. Additionally to the MIP model, we also describe in detail the structure of the address space developed for the UA server, a complex service which controls and monitors the milk processing application, and the generic sequential function chart used by the programmable logic controllers. The results indicate that the processing time is minimized, and the workload is evenly distributed between the various workstations. The overhead introduced by the architecture accounts for less than 1% of the total processing time.**

Keywords*— *service-oriented architecture, OPC UA, optimization, industrial automation, milk processing

I. Introduction

The economical environment has evolved during the last decades and years such that time-to-market requirements have become critical, and the myriad of new technologies that appear on the market have an amplifying effect on the challenges which companies face. Thus, to enable a streamlined introduction of new solutions, the production and processing systems should be both time-driven and –oriented [1].

Thus, the principal requirements which processing plants have to meet are [2]: scalability – increasing resources without stopping manufacturing, error tolerance and automated recovery from errors, dynamic integration – between departments and companies, agility – reconfigurability and flexibility, and support for both interoperable software and interoperable hardware that is heterogeneous.

The main challenges in case of device networking ecosystems are [3], [4]: widespread usage of web standard, interoperable, universal and secure access to devices, device reusability at all levels, every subsystem should be exposed as a device capable of being integrated into more complex systems, and any device should have a high-level management interface, which facilitates configuration, monitoring, fault diagnosis and maintenance, and interactions between devices should be made predictable.

In this paper we present the optimization of a milk processing application through a three-layer service-oriented industrial architecture, which follows the requirements enlisted above. The goal is to develop a generalized description of the milk processing application, which ensures a minimal processing time, which is flexible and reusable, and which handles alarm and maintenance occurrences.

II. Methods

A. Service oriented industrial architecture

We first briefly describe the service-oriented architecture initially introduced in [5]. This architecture was employed for the herein described application to automatically use the optimized milk processing plans. The architecture is based on the typical features of Service Oriented Architectures [6], namely interoperability and autonomy. The generic concept of the architecture, with its three main layers, is shown in figure 1. At the lowest levels OPC UA (Unified Architecture) servers

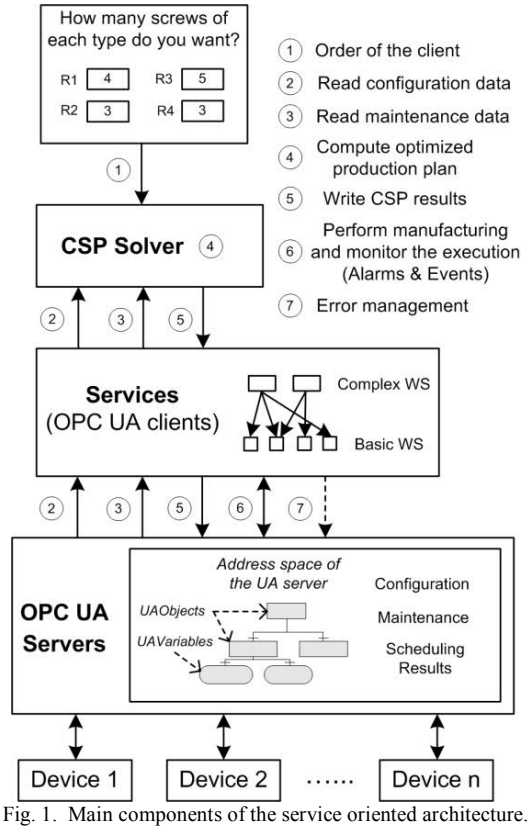

Fig. 1. Main components of the service oriented architecture.

are employed for gathering information from the control devices and from the sensors and actuators. The UA specification allows for a standardized modeling of this information and a real-time data exchange with the devices. The second layer is based on several types of software services, which ensure its adaptability and flexibility. Finally, the third layer – CSP (Constraint Satisfaction Problem) solvers – generates the optimized processing plans and schedules.

Furthermore, Figure 1 also shows how the different layers interact with each other. Once a new order is placed by a client or generated by the Enterprise Resource Planning (ERP) level (step 1), an optimized processing plan is determined based on the current state of the available stations (the current state is read from the UA server using software services – steps 2,3,4). Once the processing plan is available, a complex service is launched, which first communicates the solution to the UA server (step 5) and starts and oversees the actual processing step (step 6). If an error is generated during the processing step, the complex service also manages the resulting alarm(s) (step 7). Importantly, the address space of the UA server is automatically generated using a set of 5 algorithms [7]. Moreover, two types of services are used: basic services, which are reused in multiple scenarios and perform simple operations like writing or reading data to / from the UA server; and complex services. Besides the complex service mentioned above, other complex services are employed to extract useful information regarding the manufacturing process (e.g. extraction of Key Performance Indicators - KPIs). Related to the CSP level, the most important concepts are the model (abstract description of the manufacturing process to be optimized) and the solver (which determines the optimized processing solution). For more details on the architecture we refer the reader to [5].

B. Optimization of the milk processing plant

1) Problem description

A milk processing plant can produce three types of milk (with 1.5%, 2.5% and 3.5% fat content) on three different processing stations. Each station is equipped with an asynchronous electric motor and an inverter. The rotation velocity determines the type of milk that is being produced. In the following we consider the specific configuration in table I, where the production capacities of each station and are depicted for each type of milk (the first station can produce all three types of milk, while the other two stations can produce two types of milk). When an order is received, the goal is to produce the desired milk with a minimum of processing time. To solve the problem, a MIP model is considered, which has the goal of determining for how long each station has to work at each velocity.

Figure 2 describes the test and control structure of the application (for simplicity, the upper two levels – CSP model and services were not displayed). Inverters of type Siemens Micromaster 420 and asynchronuous eletric motors of type Bonfiglioli Riduttori 0.37kW were used. An address space corresponding to the application has been modeled for the OPC UA server. The previously introduced special adapter software solution [8] has been used to establish a connection between the UA server and the PLCs, whereas an embedded classic OPC UA client resides inside the OPC UA server.

TABLE I
PROCESSING CAPACITIES OF EACH STATION

Milk type	Station 1 [l/min]	Station 2 [l/min]	Station 3 [l/min]
1,5%	10	12	0
2,5%	8	6	10
3,5%	6	0	5

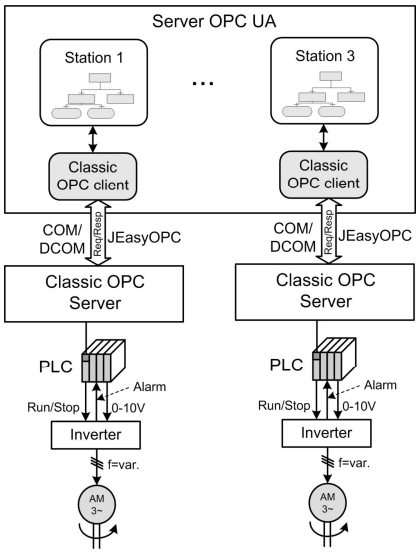

Fig. 2. Control structure of the milk processing application.

To control and monitor the inverter two binary signals are used (control signal *Run/Stop* and the feedback signal *Alarm*) together with an analog signal (0-10 V) which controls the motor velocity.

2) MIP Model

An often used subtype of CSP is linear programming [9], which is characterized by the fact that all constraints are linear. In the following we introduce a Mixed Integer Programming (MIP) model, which is based on binary and integer variables [10]. The proposed model is generic, in the sense that it can be adapted to any specific processing configuration, other than the one described in table I.

The amounts of milk ordered by the client are stored as integer variables in an array called $order[n]$, where n represents the number of different types of milk that are being obtained (its value can be changed dynamically in the model). The production capacities are stored as integer variables in two-dimensional array $cap[m][n]$, where m represents the number of stations which can be dynamically changed, similar to n. The unknowns which need to be determined by the MIP solver are the processing times on each station, for each velocity. Thus, a two-dimensional array similar to the one employed for the production capacities is being defined: $t[m][n]$ (all the variables in the array can take values between zero and infinite).

To take into account the availability of the stations (these may be unavailable during certain times due to malfunctions or maintenance actions), an additional array has been defined, called $avail[m]$, composed of binary variables.

In the following we define three constraints which ensure that sum of the processed milk quantities, separately for each type, obtained on the available stations is equal to the quantity which was ordered:

$$order[i] = \sum_{j=1}^{m} avail[j] \cdot cap[j][i] \cdot cap[j][i]50; i = 1..n \quad (1)$$

where i represents the type of milk and j represents the station.

In the following, an array is defined for storing the total processing time of each station $t\_total[m]$. A constraint needs to be defined for each variable of this array to ensure that the sum of the processing times in the array $t[m][n]$, corresponding to a certain station, is equal to the time stored in the array $t\_total[m]$:

$$t\_total[i] = \sum_{j=1}^{n} t[i][j]50; i = 1..m \quad (2)$$

By summing up the individual processing times, we ensure that the setting where different types of milk are processed on the same station is correctly taken into account (processing stages are sequential and not parallel).

Next, the objective function of the MIP model is specified. Thus, a new variable, called *t_max* is defined (the objective variable). The objective function of this model consists in the minimization of the maximum processing time on the three stations (objective function of type *makespan*). To correlate this variable with the processing time, the following constraints are defined:

$$t\_t\_total \le t\_\max; \quad i = 1..m \quad (3)$$

Thus, the last step of the model consists in minimizing the variable *t_max*.

The model is solved using the Java Linear Programming Interface (JLPI) library which is a Work In Progress Java interface developed by Siemens and which offers access to three different open-source MIP solvers: SCIP, SoPlex and GLPK.

3) Implementation Details

The address space of the milk processing application is displayed in figure 3. It is being generated by the automatic address space generation algorithms introduced in [7]. The address space has as entry point the objects *Root*, *Object*, *Devices* and *Stations*. The object called *Stations* organizes the data related to the three milk processing stations. The variables *Start* and *End*, which are also connected to the object *Devices*, represent the start and end time of a specific processing instance.

Each *Station i* object contains three objects corresponding to the three different types of milk (1.5%, 2.5% and 3.5% fat content). The corresponding motor velocity is specified underneath each of these objectives, together with the processing capacity of the station for that specific type of milk (expressed in l/min, equal to the values in table I). If a station cannot produce a certain type of milk, the capacity will be set to 0. The three motor velocities are expressed in converter units (in this case, the PLC S7-300 was employed; the analog

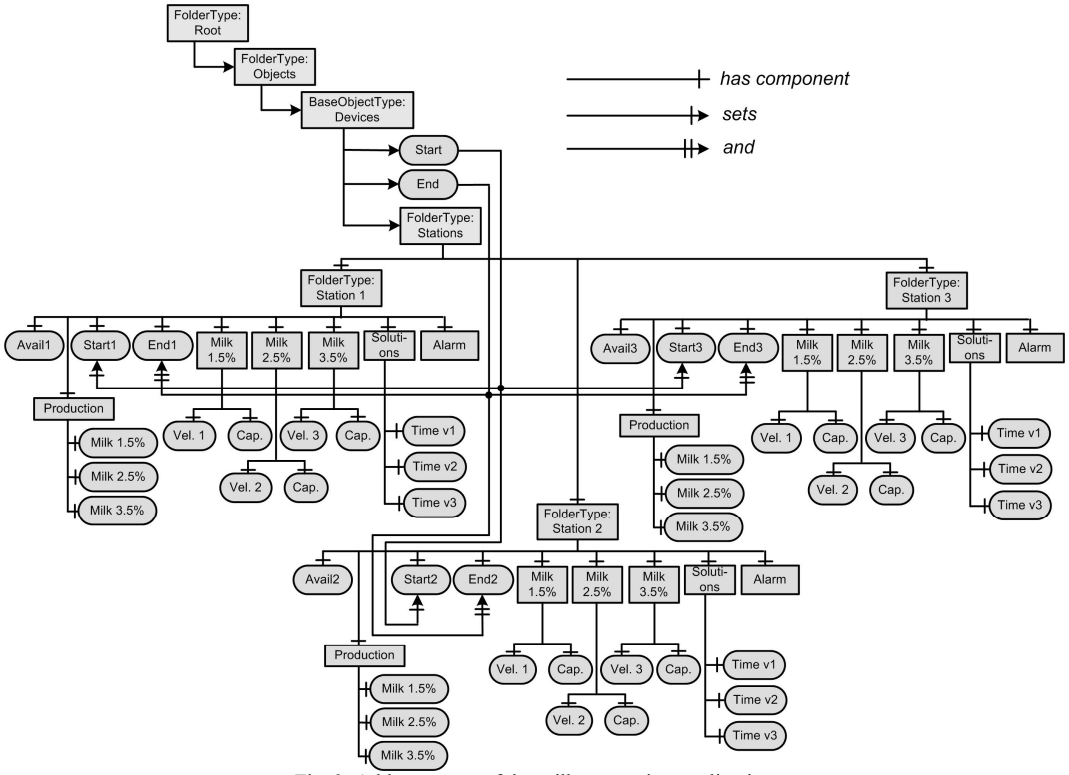

Fig. 3. Address space of the milk processing application

voltage of 0-10V, employed for controlling the inverter corresponds to an integer number – *converter units*). The structure is similar for each station, allowing for individual changes to be performed and, thus, leading to a greater flexibility during production.

Below each *Station i* object resides a *Solution* object, underneath which the optimized solution determined by the MIP solver is stored: the processing times corresponding to each velocity. These will be subsequently used by the PLCs during the production.

Another important aspect is the fact that each station has a start variable (*Start i*) and an end variable (*End i*). When the variable *Start* becomes 1, all variables *Start i* also become 1. Once a very short interval of time (100ms) has elapsed after setting these variables, they are reset to 0, to avoid a restart of the production process once the previous instance has finished.

Moreover, each station sets the variable *End i* to 1 when the processing on station *i* has finished. When all variables *End i* have become 1, the global variable *End* becomes 1. Each object/variable has a main reference and can have multiple secondary references. The references between the global variable *Start* and each *Start i* variable are secondary references called *set*. This is a hierarchical reference specifically defined for this application (it is not part of the UA specification), which links only variables and ensures that the variable which is placed at a lower level is equal to the one placed at a higher level. The references between the global variable *End* and each *End i* variable are secondary references called *and*. This type of reference can only be used to connect binary variable and enforces that the variable placed at a higher level is equal to the results of the *logical and* operation performed on the values of the variables placed at a lower level. The *End* variable is equal to 1 only if all *End i* variables are equal to 1, i.e. the production process has ended on all stations. Additionally, a binary variable *Avail i* is being introduced for each *Station i*, which specifies if the corresponding station is available for a processing instance. To count the total amount of milk which has been processed, an object called *Production* is being defined for each *Station i*, which is further employed as KPI at the ERP level.

Finally, to signal the event of an alarm, an *Alarm* object is defined for each station (it is set to 1 if the binary signal provided by the inverter becomes 1).

4) Complex service

Before solving the MIP model, the basic read variable node service is called multiple times to determine the processing times of each station and their availability. Using these values, and the values extracted from the order of the client, the MIP model is solved, and the optimized milk processing plan is determined. Subsequently, the complex service (figure 4) is called, to control and monitor the entire production process. Thus, the complex service first establishes a connection with the UA server and subscribes to the *End* variable, as well as to the three alarm nodes.

Afterwards, the MIP results received as input variables (the array $t[m][n]$) are written to the variable nodes corresponding to the three stations (the nodes *Time v1*, *Time v2* and *Time v3* of each station). Next the *Start* variable is set, and, as a result, the production process starts. During the processing, the service is able to interfere if an alarm is generated: a special service is called. If no alarm is generated, the milk production process ends as planned, the variable *End* becomes 1 and the execution of the complex service finishes.

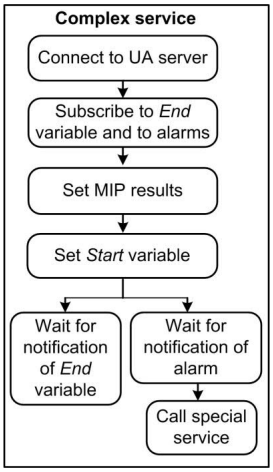

Fig. 4. Complex service used for monitoring and controlling the milk processing application.

5) PLC Control Graph

Figure 5 displays the generic sequential function chart (SFC) employed for controlling the PLC of a generic station *i*. During the initial stage (stage 1), a stop command is sent to the inverter and the variable *End i* is set to 1 to signal that at that time no processing is being performed on the station. Next, after receiving the start command (*Start i* becomes 1 as a result of setting the variable *Start* to 1 in the address space), the processing of the milk with a fat content of 1.5% starts, which means that the velocity of the motor is set to v_{1i}, which is equal to the value stored in the variable *Vel. 1* of the corresponding station in the address space. Furthermore, the signal *Run*, which starts the inverter, is set to 1. The time t_{1i} during which the motor runs at the above specified velocity is equal to the value of the variable *Time v1* of the corresponding station in the address space (as determined by the MIP model). Similarly, the processing is peformed for milk with fat content of 2.5% and 3.5% respectively (if the station does or cannot produce a certain type of milk the corresponding time is set to zero and the graph instantaneously switches to the next stage).

The processing times are expressed in minutes and the condition associated to transitions 2, 3, and 4 is of the type

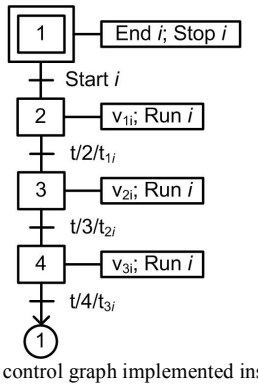

Fig. 5. Generic control graph implemented inside the PLCs.

t/y/z, where *t* – signals the fact that transition is passed after a certain amount of time has elapsed after stage *y* has been activated (*z* specifies the exact time after which the transition is passed).

III. Results

Tables II-V display the solutions obtained for differnt orders (three orders for which one type of milk has a larger quantity and one order with equal quantities of milk).

One can observe that, in each case, the execution times of the three stations are similar, which suggests that the MIP model introduced above ensures that the workload is uniformly distributed on the three workstations, so as to lead to a minimal processing time. Table VI displays the execution time of each significant module in the application for the four different orders. The results indicate that most of the processing time (over 99% in this case) is occupied by the actual processing of the milk. Operations 1 and 3 are mainly performed at the level of the services, operation 2 is peformed

TABLE II
PROCESSING TIMES AND CORRESPONDING PROCESSING QUANTITIES ON THE THREE STATIONS FOR AN ORDER OF: 1000 L MILK 1.5%, 200 L MILK 2.5%, 200 L MILK 3.5%

Station	**Time 1.5% [min] (litres [l])**	**Time 2.5% [min] (litres [l])**	**Time 3.5% [min] (litres [l])**	**Total time per station [min]**
Station 1	40.0 (400)	0.0 (0)	10.0 (60)	50.0
Station 2	50.0 (600)	0.0 (0)	-	50.0
Station 3	-	20.0 (200)	28.0 (140)	48.0
			Total execution time [min]	50.0

TABLE III
PROCESSING TIMES AND CORRESPONDING PROCESSING QUANTITIES ON THE THREE STATIONS FOR AN ORDER OF: 200 L MILK 1.5%, 1000 L MILK 2.5%, 200 L MILK 3.5%

Station	**Time 1.5% [min] (litres [l])**	**Time 2.5% [min] (litres [l])**	**Time 3.5% [min] (litres [l])**	**Total time per station [min]**
Station 1	2.0 (20)	26.0 (208)	30.0 (180)	58.0
Station 2	15.0 (180)	42.0 (252)	-	57.0
Station 3	-	54.0 (540)	4.0 (20)	58.0
			Total execution time [min]	58.0

TABLE IV
PROCESSING TIMES AND CORRESPONDING PROCESSING QUANTITIES ON THE THREE STATIONS FOR AN ORDER OF: 200 L MILK 1.5%, 200 L MILK 2.5%, 1000 L MILK 3.5%

Station	**Time 1.5% [min] (litres [l])**	**Time 2.5% [min] (litres [l])**	**Time 3.5% [min] (litres [l])**	**Total time per station [min]**
Station 1	2.0 (20)	1.0 (8)	90.0 (540)	93.0
Station 2	15.0 (180)	32.0 (192)	-	47.0
Station 3	-	0.0	92.0 (460)	92.0
			Total execution time [min]	93.0

TABLE V
PROCESSING TIMES AND CORRESPONDING PROCESSING QUANTITIES ON THE THREE STATIONS FOR AN ORDER OF: 700 L MILK 1.5%, 700 L MILK 2.5%, 700 L MILK 3.5%

Station	**Time 1.5% [min] (litres [l])**	**Time 2.5% [min] (litres [l])**	**Time 3.5% [min] (litres [l])**	**Total time per station [min]**
Station 1	2.0 (20)	1.0 (8)	90.0 (540)	93.0
Station 2	15.0 (180)	32.0 (192)	-	47.0
Station 3	-	0.0	92.0 (460)	92.0
			Total execution time [min]	93.0

TABLE VI
EXECUTION TIMES AT DIFFERENT LEVELS OF THE ARCHITECTURI FOR THE ORDERS IN TABLES II-V

Nr.	Operation	Order 1		Order 2		Order 3		Order 4	
		Time [s]	Perc. [%]	Time [s]	Perc. [%]	Time [s]	Perc. [%]	Time [s]	Perc. [%]
1	Reading configuration data	2.61	0.09	2.65	0.07	2.64	0.05	2.60	0.05
2	Computing MIP solution	0.32	0.01	0.33	0.01	0.32	0.01	0.34	0.01
3	Writing of MIP solution into the UA address space	2.32	0.08	2.33	0.07	2.31	0.04	2.34	0.04
4	Production process	3000.3	99.8	3480.4	99.8	5400.2	99.9	5580.3	99.9

at the CSP level, while operation 4 is performed at the service, UA server, and control device level.

IV. Conclusions

In this paper we have presented the optimization of a milk processing application through a service-oriented industrial architecture composed of three layers. A generic model of the processing application has been developed, which leads to a minimal production time. The tests run for four different orders have confirmed that the workload is evenly distributed and the overhead introduced by the requirement of solving a CSP model is minimal: the actual processing step still occupies over 99% of the total processing time. Typically, the definition of a CSP model represents a trade-off between the model and performance. A small execution time may be obtained if the CSP model is specific for a certain plant setup, but the maintenance time will considerably increase when changes are required.

Conversely, if the models are generic, the time required to solve the model increases, but the model is much more flexible. In this specific case, we have decided to develop a more generic model, since the time required by the solver to find the optimal solution is orders of magnitude smaller than the actual processing time: both the number of processing stations and the types of milk to be produced can be modified without having to change the CSP model.

As for future work, we plan to evaluate other approaches for solution optimization, that have been used successfully in other research areas [11], [12].

Acknowledgment

This work was supported by a grant of the Romanian National Authority for Scientific Research and Innovation, CCCDI – UEFISCDI, project number ERANET-FLAG - RoboCom++ (2), within PNCDI III.

References

[1] M. Surridge, S. Taylor, D. De Roure, and E. Zaluska, "Experiences with GRIA – industrial applications on a web services grid," *1st Inter. Conf on e-Science and Grid Computing*, Melbourne, July 2005, pp. 98-105.

[2] F. Jammes, and H. Smit, "Service oriented paradigms in industrial automation", *IEEE Trans. on Industrial Informatics*, vol. 1, pp. 62-70, Feb 2005.

[3] H. Bohn, A. Bobek, and F. Golatowski, "SIRENA - service infrastructure for real-time embedded networked devices: a service oriented framework for different domains," Proc. of the Inter. Conf. on Networking, Inter. Conf. on Systems, and Inter. Conf. on Mobile Communications and Learning Technologies, April 2006, pp. 43-47.

[4] A. Sasa, M.B. Juric, and M. Krisper, "Service-oriented framework for human task support and automation," *IEEE Trans. on Industrial Informatics*, vol. 4, pp. 292-302, Nov. 2008.

[5] A. Girbea, C. Suciu, S. Nechifor, and F. Sisak, "Design and implementation of a service-oriented architecture for the optimization of industrial applications", *IEEE Trans. on Industrial Informatics*, vol. 10, pp. 185-196, Feb. 2014.

[6] T. Cucinotta, A. Mancina, G.F. Anastasi, G. Lipari, L. Mangeruca, R. Checcozzo, and F. Rusina, "A real-time service-oriented architecture for industrial automation," *IEEE Trans. on Industrial Informatics*, vol. 5, pp. 257-266, Aug. 2009.

[7] A. Girbea, S. Nechifor, F. Sisak, and L. Perniu, "Efficient address space generation for an OPC UA server", *Software-Practice and Experience*, vol. 42, pp. 543-557, 2012.

[8] A. Girbea, S. Nechifor, F. Sisak, and L. Perniu, "Design and implementation of an OLE for process control unified architecture aggregating server for a group of flexible manufacturing systems", *IET Software*, vol. 5, no. 4, pp. 406-414, Aug. 2011.

[9] T. Sawik, *Scheduling in supply chains using mixed integer programming*. Hoboken, NY: Wiley, 2011.

[10] Y. Chu, F. You, "Integration of scheduling and control with online closed-loop implementation: Fast computational strategy and large-scale global optimization algorithm", *Computers & Chemical Engineering*, vol. 47, no. 20, pp. 248-268, 2012.

[11] I.A. Tache, "Vessel Segmentation of Coronary X-ray Angiograms", Proc. of the 20th International Conference on System Theory, Control and Computing, October 2016, pp. 727-731.

[12] I.A. Tache, "Vessels Enhancement in X-ray Angiograms", Proc. of the IEEE International Conference on e-Health and Bioengineering, November 2015,