Int j simul model 6 (2007) 3, 165-172
ISSN 1726-4529 Professional paper

INTEGRATION OF SIMULATION SOFTWARE ARENA WITH FMS CONTROL SYSTEM

Slota, A. & Malopolski, W.
Cracow University of Technology, Production Engineering Institute,
Al. Jana Pawla II 37, 31-864 Cracow, Poland
E-Mail: slota@mech.pk.edu.pl; malopolski@ mech.pk.edu.pl

Abstract
Simulation software enables to build and process FMS models to analyse different aspects of the system operation. Such models, verified in simulation experiments, are ready to be used for FMS control to calculate control decisions. The idea of using simulation software Arena to control FMS is presented in the paper. Basic characteristic of modelling and simulation in Arena is presented. Arena real time module, which enables communication with external applications when the model is processed, is characterized. Required configuration options for Arena and the model are presented. To integrate Arena with local controllers of the FMS a computer program is designed. The proposed solution is implemented and verified for a real system.
(Extended paper from the 17th International DAAAM Symposium, Vienna, Austria, 8-11 November 2006.)

Key Words: Modelling, Simulation, FMS Control

1. INTRODUCTION

Rapid development of highly automated machine tools, especially in the area of control, gives an opportunity to build truly Flexible Manufacturing Systems (FMS). Such systems are designed for time and cost effective manufacturing products in small batches. Products may be processed in FMS according alternative routes. During FMS operation the production undergoes changes: new products and new variants of products are introduced, production capacity fluctuates. Some unexpected events may also occur – machines may become temporarily unavailable because of a failure, new machines may be added to replace unavailable ones or to extend system capabilities [1]. Such events, referred to as disturbances, may substantially affect effectiveness of the system [2]. When a disturbance occurs its impact should be assessed and, if it is necessary, new control policies ought to be worked out and proved correct before changes are incorporated into the real systems.

Due to complex structure and dynamic character of real manufacturing systems the way the FMS works is often analysed with the use of simulation techniques. Though simulation does not provide full insight into the system operation, like analytical analysis does, it is used to verify system's behaviour under different conditions. Simulation is used in the areas of FMS design and reconfiguration [3], scheduling [4, 5, 6] and performance evaluation [7, 8]. Simulation is an execution of a virtual process which corresponds to operation of the real system under defined conditions. Data crucial for analysis of FMS operation are included in the model which is processed during simulation experiment. Time necessary to run a computer simulation experiment depends on the complexity of the model, effectiveness of simulation software and computer hardware and the horizon of simulation experiment. Thus performance of different control strategies, taking into account operation condition (varying production volume, level of disturbances), may be quickly explored to find the one which is acceptable.

For FMS control some kind of a mathematical model of the system is also required. Such model, processed in real time, is used to calculate control decisions. In this case the model evolves along with changes in the real system. This is done by exchanging messages between the control software processing the model and local controllers of the system. Thus it comes natural, in the first step, to make use of the model proved to be correct in simulation for control of a real system. The second step is to use simulation software, which processes the model during simulation, for processing the same model during control. The above goals may be accomplished by building user simulation-control software [9, 10, 11]. Such solutions require great amount of work to design, develop, test and implement software which, in most cases, is designed for a given system and their limited openness and flexibility makes it difficult to apply them to other systems. Available commercial simulation packages are listed and characterized in [12][12]. Some simulation packages have built in mechanisms for integration with external applications, which may be used for communication between simulation software and local controllers of FMS. The authors decided to work out and implement FMS operational control with the use of Rockwell simulation software Arena.

2. ARENA SOFTWARE

2.1 Modelling and simulation

Arena simulation software is a tool for modelling and simulation discrete event systems [13]. It provides an intuitive environment for model creation. The model consists of two types of modules: *Flowchart* modules and *Data* modules. Flowchart modules are placed in the model window. Connections between these modules describe the logic of the process and define entities flow between modules. Data modules are presented in the form of a spreadsheet. They are used to supplement the model with quantitative data crucial for simulation like process/transport times, resource requirements, processes' schedules, etc. Defined models are recorded with the use of SIMAN language [14] and then simulated. After simulation reports on different criteria (such as: usage of resources, length and waiting time in queues, time entities spend in system) are generated to assess operation of the system. Arena's additional applications help to define input data for simulation (*Input Analyzer*), to manage different model configurations (*Process Analyzer*), to analysis of simulation results (*Output Analyser*) and to find optimal model configuration (*OptQuest*).

2.2 ARENA Real Time mode

Arena real time (RT) mode allows a model to be run in execution mode. In this case Arena coordinates processing of the model with real-world processes by exchanging messages with an external application. This feature may be used for monitoring and control of real systems [13]. To run Arena in execution mode the model has to include definitions of messages which are to be sent by Arena to external processes. Data and format of messages are defined by *TASKS* elements. If a message should be sent to external process when an entity enters a module, expression defining *TASKID* has to be entered for velocity or duration field. The format is *TASKID (Value, TaskID),* where *Value* defines duration or velocity for simulation purposes and *TaskID* points which *TASKS* element defines data and format of the message. Fig. 1 shows sample definitions of *TASKS* element and *TASKID* for *PROCESS* module.

Messages may be sent by Arena when an entity enters the following modules: *TRANSPORT*, *PROCESS*, *ENTER*, *LEAVE*, *DELAY*, *ROUTE* and *MOVE* (*TASKID* may be assigned to these modules). To reply to the Arena message external process uses *TGID* defined in *TASKS* element. Arena may also receive and process messages which are not

associated to any message sent by Arena. Such messages are referred to as *Unsolicited messages*.

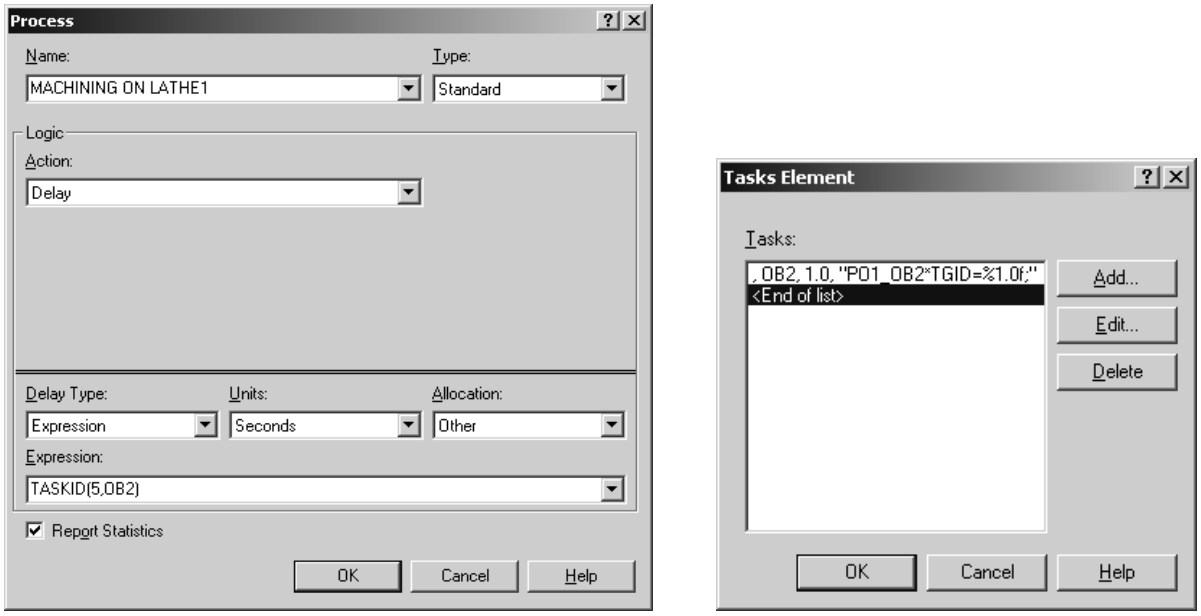

Figure 1: Arena dialog boxes for PROCESS module and TASK definition.

Arena communicates with external processes through communication library (dynamic linked library) *RTDLL.dll*. The library implements socket communication mechanism. To enable exchanging messages library *RTDLL.dll* should be loaded, option *Run in Execution Mode* should be checked and value for *Advance Simulation Time Using a Real Time Factor* should be set to 1 (Fig. 2).

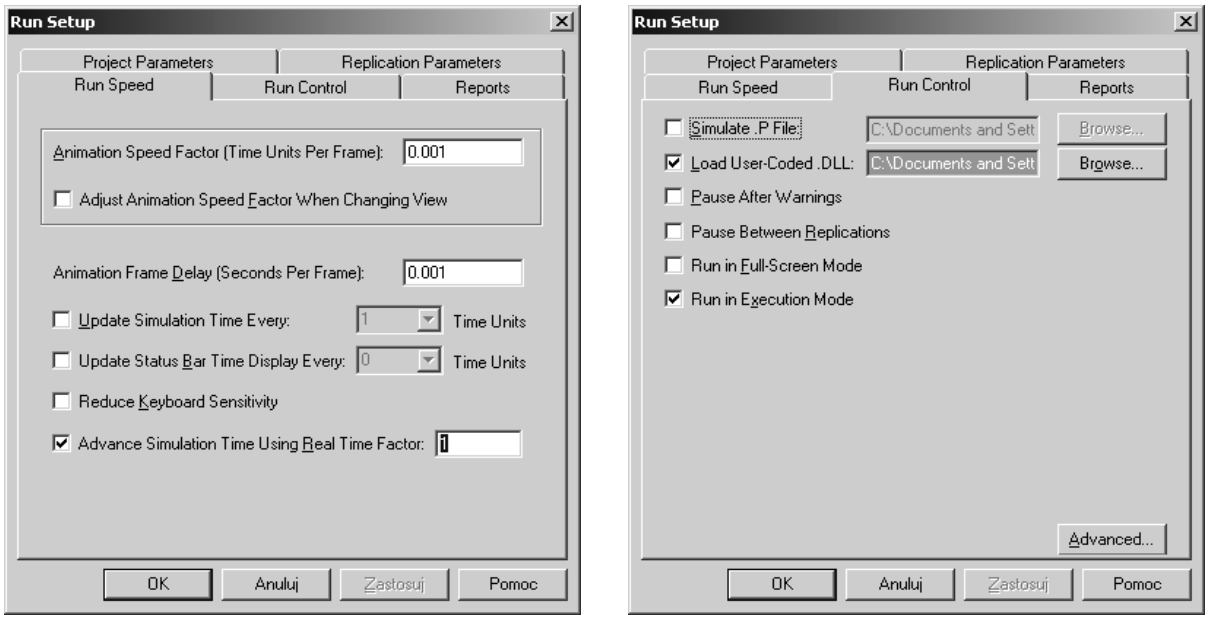

Figure 2: Arena RT settings.

The first step of running Arena model in execution mode is establishing a connection with an external process. Arena goes into a listen state and waits for connection request. After a correct connection request has occurred it is accepted and socket communication is established. Any time Arena model needs to send a message to an external process

writeIPCQueue function, which sends messages through the socket, is called. When an external process sends a message to Arena *readIPCQueue* function, which reads data form the socket, is called. These functions are implemented in *RTDLL.dll* provided with Arena.

3. AN IDEA OF FMS CONTROL WITH ARENA

Arena RT provides communication mechanism and tools for definition of messages (data, format and when the message should be sent). These messages will be used as task commands sent from Arena to local controllers of the system to execute activities. Arena may receive reply messages from local controllers which will be used as confirmation of activities' execution. When an entity enters a module with assigned *TASKID* a command to execute an activity is sent to local controllers. The entity is held in the module for a defined period of time (during simulation) or until a reply message confirming activity execution is received (during control). The diagram of the Arena based FMS control system is presented in Fig. 3.

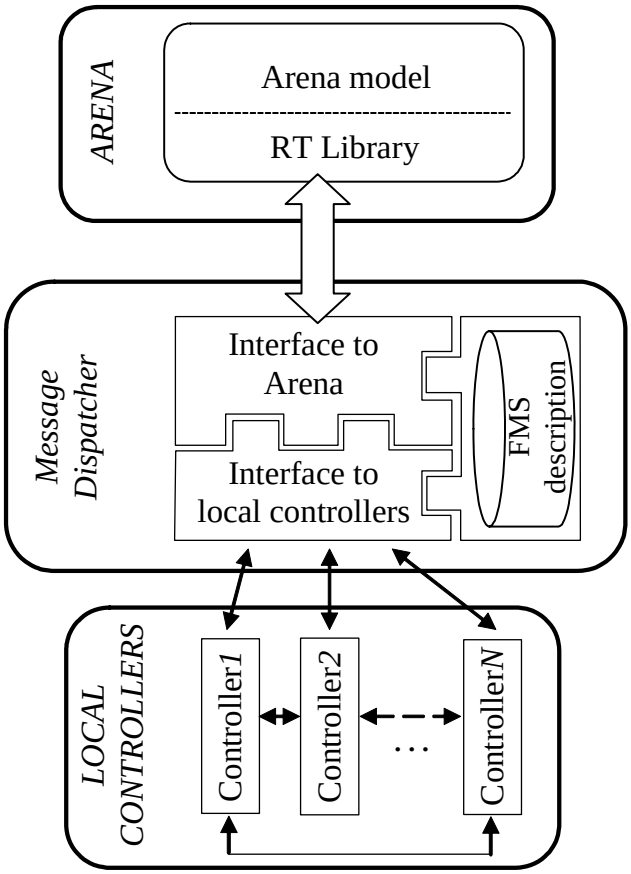

Figure 3: Structure of Arena based FMS control system.

To integrate Arena with local controllers of FMS computer program Message Dispatcher is introduced. Message Dispatcher executes the following tasks:

- receives commands of activity execution from Arena, translates them into the format required by local controllers and dispatches them to local controllers which are responsible for the activity execution,
- receives messages confirming activity execution from local controllers, translates them into the format required by Arena and sends them back to Arena.

Data necessary to translate and transfer messages between Arena and local controllers are stored in the module FMS description.

4. IMPLEMENTATION AND VERIFICATION

4.1 EMCO system description

The proposed solution of using Arena for FMS control is implemented for an educational production system EMCO in the laboratory of Production Engineering Institute at Cracow University of Technology.

The system consists of two lathes EMCO COMPACT 5PC and robot MITSUBISHI which transports workpieces between a small storage and the machines. The layout and control structure of the system is shown in Fig. 4.

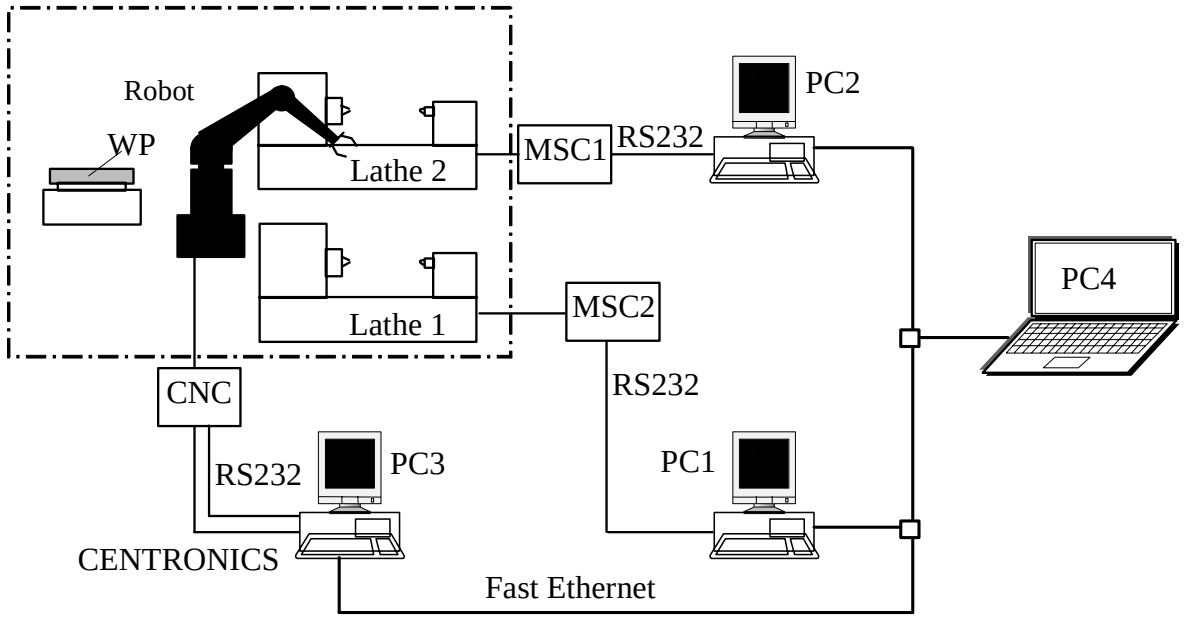

Figure 4: Diagram of EMCO control system [9].

The main parts of local controllers, taking into account communication, are the controllers' software interfaces running on PC1, PC2 and PC3 computers. They use Distributed Component Object Model technology (DCOM) technology [15] to communicate with each other and with higher level controller. Message Dispatcher runs on PC1 and communicates with these software interfaces of local controllers. Arena runs on PC4.

4.2 Implementation of Message Dispatcher

Message Dispatcher is an application coded in C++ with the use of Microsoft provided MFC library. It is designed to run under Windows NT/XP operating system. To exchange messages with local controllers Message Dispatcher uses DCOM technology.

FMS description contains a list of local controllers of the system *Contr_list* and a list of activities executed in the system *Act_list*. Each controller in *Contr_list* is defined by the following data: *c_name* – name of the controller, *c_guid* – unique identifier of the controller used in DCOM, *comp_name* – name of the computer which runs the controller, *c_status* – a flag indicating if the controller is connected to Message Dispatcher. Each activity in *Act_list* is defined by data: *Arena_act_name* – name of the activity defined in Arena, *Contr_act_name* – name of the activity defined in local controllers, *act_ID* – activity identifier generated by Arena, *contr_list* – list of controllers which are responsible for activity execution, *exe_ contr_list* - list of controllers which are currently executing the activity.

In the initial stage Message Dispatcher establishes connections with Arena and with local controllers of FMS included in *Contr_list*.

Window of running Message Dispatcher with a list of messages exchanged between Arena and local controllers of EMCO system is presented in Fig. 5.

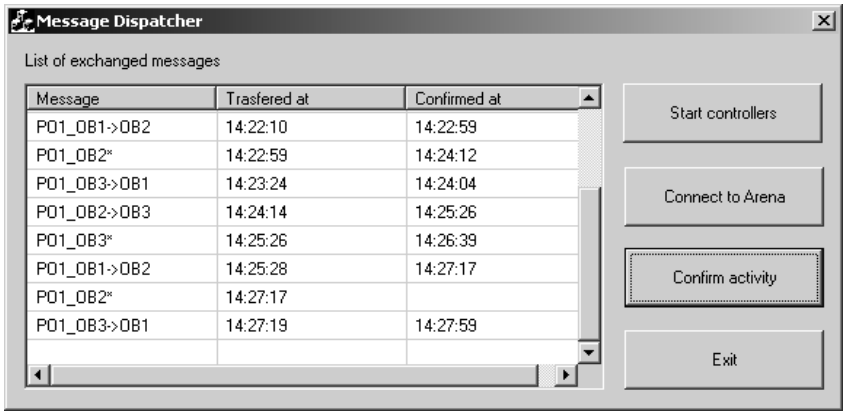

Figure 5: Message Dispatcher window.

Operation of Message Dispatcher is driven by messages received asynchronously from Arena and local controllers. Tasks executed by Message Dispatcher when a message is received are presented in Fig. 6.

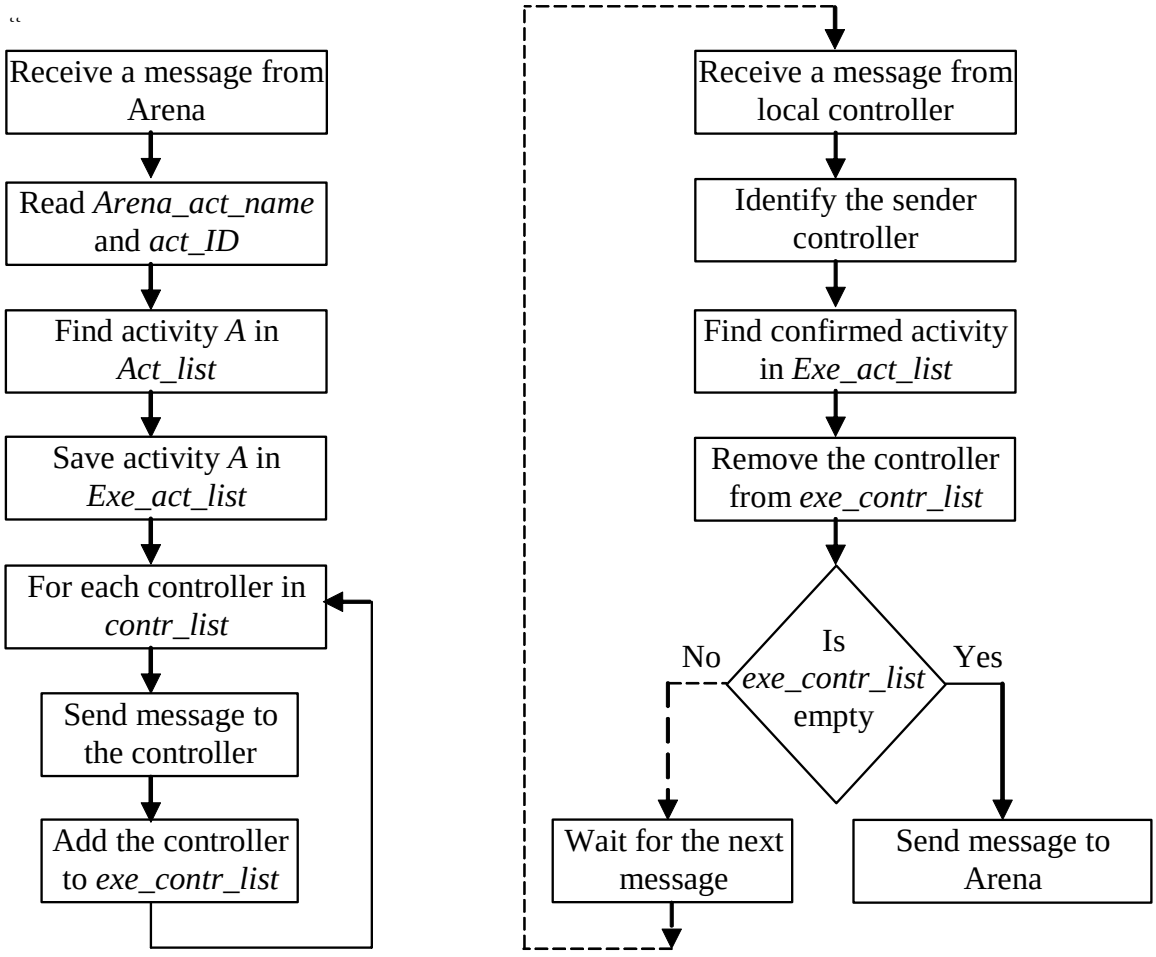

Figure 6: Diagram of Message Dispatcher operation.

4.3 Arena model of EMCO system

Arena model of EMCO system is presented in Fig. 7. In the model three stations are defined: TABLE, LATHE1 and LATHE2. Routing is added to each generated workpiece (LATHE1 and then LATHE2) and workpieces are moved to TABLE station. Then single workpieces are processed in the system according defined routes.

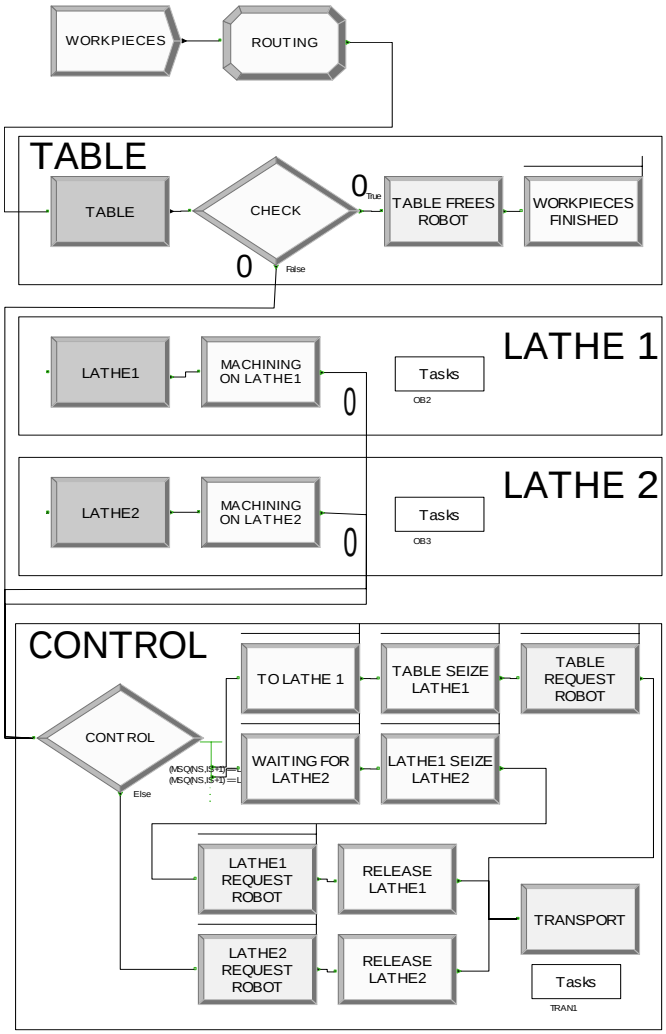

Figure 7: Real time Arena model of EMCO system.

For communication purposes three tasks elements are defined in the model. They are assigned to process modules: MACHINING ON LATHE1 and MACHINING ON LATHE2 and to transport module TRANSPORT. Tasks elements define data and format of messages which Arena sends to Message Dispatcher. Initial and final positions of a workpiece for different transport activities are calculated in CONTROL part of the model on the basis of the workpiece's current state and its route.

5. CONCLUSION

Building FMS control software is a laborious, time consuming and error prone process. Simulation is a basic technique for analysis and verification of FMS control systems. The possibility of using available simulation software and verified simulation models for FMS control eliminates all the work necessary to build special control systems and adapt

simulation models for control. What is necessary is to integrate simulation software with existing local controllers of FMS. In the paper FMS control system based on Arena simulation software is presented. To enable exchanging messages between Arena and controllers of FMS Message Dispatcher is designed and then implemented and verified for EMCO system. Since local controllers of FMS (CNC, PLC, PC based controllers) may require different communication protocols and data format to exchange information with other applications implementation of *Interface to local controllers* module of Message Dispatcher (Fig. 3) is a dedicated solution. The future work will concern restructuring Message Dispatcher so that it is more flexible. FMS description should be stored in an external file (database, or text file). Interface to local controllers should be implemented separately, for example as a dynamic linked library – communication library. These changes will enable using Message Dispatcher application for different systems just by replacing database file and communication library. Message Dispatcher and model of FMS ought to be developed to handle error messages caused by system devices and communication failures.

REFERENCES

[1] Valckenaers, P. (2000). Analysis and Evaluation of change and disturbances in industrial plants, *WP1DisseminationReport*, MASCADA Project, ESPRIT LTR 22728, K.U. Leuven, PMA

[2] Ylipää, T. (2002). Correction, prevention and elimination of production disturbances, *PROPER project description*, Department of Product and Production Development (PPD), Chalmers University of Technology, Gothenburg

[3] Lalic, B.; Cosic, I.; Anisic, Z. (2005). Simulation based design and reconfiguration of production systems, *International journal of simulation modelling*, Vol. 4, No. 4, 173-183, doi:10.2507/IJSIMM04(4)2.047

[4] Jain, A.; Jain, P. K.; Singh, I. P. (2004). An investigation on the performance of dispatching rules in FMS scheduling, *International journal of simulation modelling*, Vol. 3, No. 2-3, 49-60

[5] Joines, J. A.; Barton R. R.; Kang, K.; Fishwick, P. A. (2000). A simulation test-bed to evaluate multi-agent control of manufacturing systems, *Proceedings of the 2000 Winter Simulation Conference*

[6] Chick, S.; Sánchez, P. J.; Ferrin, D.; Morrice, D. J. (2003). Simulation-based scheduling for dynamic discrete manufacturing, *Proceedings of the 2003 Winter Simulation Conference*

[7] Pierzchala, W. (1999). Planowanie zadań produkcyjnych w oparciu o wirtualny proces wytwarzania, *Postępy Technologii Maszyn i Urządzeń*, Vol. 23, No. 4, 71-90

[8] Jain, P. K.; Fukuda, Y.; Komma, V. R.; Reddy, K. V. S. (2006). Performance modelling of reconfigurable assembly line, *International journal of simulation modelling*, Vol. 5, No. 1, 16-24, doi:10.2507/IJSIMM05(1)2.049

[9] Slota, A. (2003). Application of Object Observable Petri Nets in simulation and operational control of discrete manufacturing systems, *Proceedings of the 14th International DAAAM Symposium*, 431-432

[10] Zając, J. (1999). Modelling Manufacturing Control System: Multi-Agent Approach. *Postępy Technologii Maszyn i Urządzeń*, Vol. 23 No. 4, 137-153

[11] Cyklis, J.; Zając, J.; Słota, A. (2004). Models of manufacturing system for simulation and control, *Manufacturing Engineering*, Vol. 4, No. 3. 10-15

[12] Klingstam, P.; Gullander, P. (1999). Overview of simulation tools for computer-aided production engineering, *Computers in Industry*, Vol. 38, No. 2, 173-186

[13] Kelton, W. D.; Sadowski, R. P.; Sadowski, D. A. (2002). *Simulation with Arena*, McGraw-Hill Companies, ISBN 0-07-112239-7, New York

[14] Pegden, C. D.; Shannon R. E.; Sadowski R. P. (1995). *Introduction to Simulation Using SIMAN*, McGraw-Hill, Inc.

[15] Zajac, J. (1998). Interobject communication in distributed manufacturing using COM technologies, *Proceedings of the 9th International DAAAM Symposium*, 513-514