FRI-ONLINE-1-CCT1-09

AN ARCHITECTURAL SOLUTION FOR MEDIATION AND OPTIMIZING WORK EFFICIENCY BETWEEN THE ENTERPRISE RESOURCE PLANNING AND THE TRANSPORT CONTROL SYSTEMS[9]

Iliya Draganov, PhD Student
Department of Computer Science,
University of Ruse
Tel.: +359 883 476542
E-mail: idraganov@uni-ruse.bg

Svetlana Stefanova, PhD
Department of Computer Science,
University of Ruse
Phone: +359 888 309579
E-mail: sstefanova@ecs.uni-ruse.bg

Abstract: *Businesses that are using ERP (Enterprise Resource Planning) systems and such that control internal logistics are looking for compatible software solutions that make it easier to optimize and synchronize their work. This publication offers an architectural software solution to mediate between ERP and AGV (Automated Guided Vehicle) / TC (Transport control) system in order to optimize the process of executing transport orders. The presented solution describes not only the internal organization and structure, but also a certain functionality for external connectivity to other software, which makes it an optional, but increasing the efficiency of the mediator.*

Keywords: *AGV, ERP, Efficiency, Software Mediator*

INTRODUCTION

Businesses engaged in manufacturing or transportation most often use an ERP system to monitor and control their resources. More and more of the mentioned companies also use control systems for their automated vehicles - TC / AGV systems. It is preferable for the two systems to be able to communicate and exchange information with each other for the complete and successful automation of work processes. One of the approaches to achieve such a goal is an agreement between the software companies developing the two types of systems to build a standard in the communication between them. Currently, individual companies developing ERP systems offer solutions that work only with certain products of companies developing TC / AGV systems. On the other hand, TC / AGV systems do not always offer settings, or at least not easily adjustable ones, that can be adapted to the specific needs or requirements of the client.

EXPOSITION

Proposed solution

This paper proposes a software solution that takes care of the communication between the two types of systems, facilitating the adjustment between them and acquiring some of their functions. As modern ERP systems are extremely rich in functionality and have proven their stability and reliability over time, a better line of work is to facilitate the operation of the TC / AGV system. Its main function is the direct control of robots / AGVs at any time during the work process, which is expressed in the issuance of commands to move from point to point or action on the spot within the specified route. Currently, most TC / AGV systems are responsible for preparing the work schedule, but this is not always effective, especially in conditions of operation with changing parameters, i.e.

[9] Докладът е представен на сесия FRI-ONLINE-1-CCT1 на 13 ноември 2020 г. с оригинално заглавие: AN ARCHITECTURAL SOLUTION FOR MEDIATION AND OPTIMIZING WORK EFFICIENCY BETWEEN THE ENTERPRISE RESOURCE PLANNING AND THE TRANSPORT CONTROL SYSTEMS

for full automation with optimal results it is necessary to take into account some additional factors when preparing the schedule.

If the offered middleware is divided into separate context modules, then it can be implemented as a distributed system using limited hardware resources to cope with its tasks. As a result, fewer hardware resources lead to lower maintenance costs. Another important point in creating such an intermediate system is the interface, which should make it easier for the user to adjust settings. Consequently, the additional costs of maintenance by persons specialized in this field will be eliminated.

Architecture of the proposed solution

A database with clearly distinguishable tables by purpose should be the basis in the overall architecture. The business logic of the software must have constant access to it and serve as an intermediary for the external software that will send or request information. It can be defined as a distributed system, whose functionalities should be in separate bound contexts, i.e. in modules focused on a small set of functions, processing data only for a specific purpose or related to a specific operation, communicating with each other through certain rules or protocols.

In modern software, the implementation of such architecture most often happens through microservices. A microservice performs contextually defined operations on data received by it, while being connected to other microservices and / or external software and is necessarily process-independent of them. Each of the microservices should be responsible for as few and as simple functions as possible, but necessarily related to a specific object or action. It is good practice for two microservices to use different tables and never the same one within the database.

Figure 1 shows the architecture of the proxy software. The business logic is represented by the abstractly defined set of Microservices, whose functions for communication with external systems and the internal database are clearly distinguished. Individual microservices will take care of the communication with the ERP system, the communication with the TC / AGV system, the connection to the database and the provision of the REST API interface for other external software. The listed functionalities do not limit the set of microservices and other ones belong to it, part of the work of the intermediary software. It is important to mention that these microservices also communicate with each other through an internal isolated common communication channel. Through it, they send signals or messages when an event occurs in one of them, as well as receive such from other internal or external sources for the software. Most often the realization of the communication between them is through software providing an event bus.

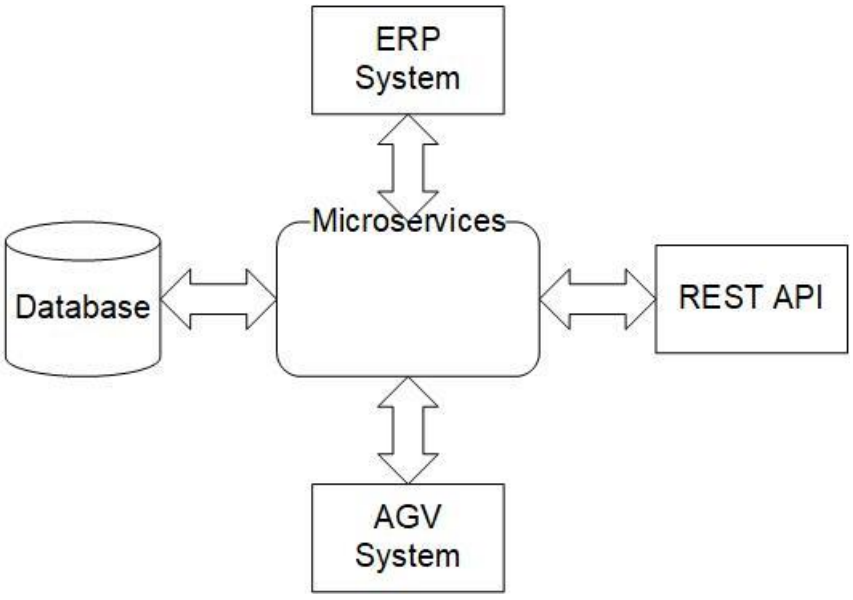

Fig. 1. Abstract architecture of the intermediate software

Modules description

Each microservice is represented by an intermediary software module. Each of the modules must be functionally independent of the others, so that in case of a problem it does not interfere with the entire working process of the system. Communication between them takes place through messages in a strictly defined format, and the content of the communicated data depends on the context of the "sender". Messages have topics and can be sent to a specific module, or broadcasted over the entire bus. When broadcasting on the entire bus, the message whose subject coincides with the context of the module function will be processed by the respective microservice. The main microservices that need to be provided are: order processing, robot status processing, collecting virtual map information, processing user settings, calculating routes, calculating the possibility of successful execution - estimator, pairing an order with a robot, orders arrangement. The corresponding modules for each of the services are as follows:

AGV / robot module

Retrieves and stores information about all robots in the TC / AGV system, providing it to other modules via the event bus, and to external software through the REST API interface. The messages that it sends on the bus reflect changes in the status of the robots. This module also stores logs related to its operation

Virtual card module

Retrieves information about the work environment virtual map, most often provided by the TC / AGV system or from a separate file. The sent messages are in response to others that require information about a point or path of the virtual map. The other modules are notified in case of change of the virtual card. Additional functionality is the provision of information to external software via the REST API interface

Module for storing user settings

Accepts, stores and provides user settings on request from another module. It has a REST API interface for interaction with the user frontend interface. Both the virtual map module and this one do not keep logs, as they have primitive functionality and are not closely connected to an external system for the intermediate software.

Route calculation module

This module is entirely internal to the software and communicates with the others only via the event bus. Its main function is to calculate the route between the set points in an order. Additionally, it sends messages to request information about the virtual map and user settings, and in some cases about robots. Once it calculates a route, the data for it is saved, as well as data about the path between any two separate points. Logs are kept of the events related to its activity, for the purpose of future analysis of the performed work.

Estimator module

P Also a completely internal module that calculates and provides data related to the execution of orders by robots - enough battery to execute, time to complete an order, average execution time based on collected previous results, etc. It plays an essential role in choosing the right robot to complete the next pending order set by the ERP system. Messages are sent in response to other modules, which must have information on whether and to what extent any of the robots is in a minimum acceptable condition to complete a specific order. All data for successfully completed orders is saved in order to improve the efficiency of the robots by optimizing the schedule. Information is requested from the user settings module to adapt the results according to customer requirements. After successful completion of an order subject to the set conditions, it again saves the results as a combination of the optimally achieved results and set requirements for future possible use of the same calculation. In addition to the data presenting the results of the calculations, it also keeps logs of events related to its activity.

Pairing module

An entirely internal module, that sends messages to collect the necessary information from the other modules and to form an information pair of an order and the most suitable of the available robots for its execution. Accepts messages from the order module when a new one appears and asks the robot module for available AGV, after which it requests route calculation and approximate results for the different combinations of order and robot. The complete information is sent to the next module in the chain of the robot order selection process. Logs are kept for all pairings in order to analyse the results in the future.

Orders arrangement module

Prepares a new order arrangement according to the available robots and customer preferences. This occurs after the pairing module informs about a new order-robot pair. The message for the new order arrangement is then sent to the order module. This module is not entirely internal so that it can be monitored and, if necessary, adjust the order of execution orders, which happens via the REST API interface. Logs of the work performed are kept in order to analyze the correct performance of the functions.

CONCLUSION

The existence of an intermediary software to take care of the communication between ERP and AGV systems solves the problems with the lack of a standard for integration of the exchanged data, as well as with the user settings. This would make it easier for users to achieve flexibility and fluidity in configurating the work process that includes the above-mentioned types of systems. The distributed type of architecture allows for easier expansion and modification of the software, and a fast and simple method of running diagnostics on a problematic module. Details of the proposed solution's realization will be the subject of a subsequent publication.

REFERENCES

Fielding, R. T. (2000), *Chapter 5: Representational State Transfer* (*REST*), Architectural Styles and the Design of Network-based Software Architectures (PhD). University of California, Irvine

openTCS: User's Guide, https://www.opentcs.org/docs/5.0/user/opentcs-users-guide.html (Accessed on 30.08.2020)

openTCS: Developer's Guide, https://www.opentcs.org/docs/5.0/developer/developers-guide/opentcs-developers-guide.html (Accessed on 30.08.2020)

Wolff, E. (2016), Microservices. Flexible Software Architectures.