Available online at www.sciencedirect.com

ScienceDirect

Procedia Engineering 100 (2015) 1572 – 1581

Procedia Engineering

www.elsevier.com/locate/procedia

25th DAAAM International Symposium on Intelligent Manufacturing and Automation, DAAAM 2014

Aspects of Smart Manufacturing Via Agent-Based Approach

Aleksandr Fedorov, Egor Goloschchapov, Oleg Ipatov, Vyacheslav Potekhin, Viacheslav Shkodyrev, Sergey Zobnin*

Saint-Petersburg State Polytechnic University in Russia, 29, Polytechnicheskaya st., St.Petersburg, 195251, Russia

Abstract

This paper focuses on three different aspects of a multi-agent approach implementation applied in smart manufacturing. Information systems and their internals are considered as a domain of the research. Among a variety of approaches and implementation techniques the following and presented: networks of self-organising heterogeneous agents are used to address problems of increasing server loads, AI techniques (POMDPs) are used to ensure highest possible availability via integrated reasoning, special optimization routines are designed to increase performance of historical access to time series data. Described approaches aim on advanced scalability, high fault-tolerance and low latency execution of a resulting system respectively. Each technique defines a mathematical model that is used to optimize corresponding criteria. For each technique implementation peculiarities and details are included.

© 2015 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).
Peer-review under responsibility of DAAAM International Vienna

Keywords: smart manufacturing; agent-based approach; peer-to-peer; fault-tolerance; latency; architecture

1. Introduction

Smart manufacturing is considered as one of the promising technologies of integrated automation and robotization of big industrial systems nowadays. Modern trends of industrial systems development show that such a development is tightly connected with adoption of artificial intelligence approaches and attempts to increase performance, especially in case of complex manufacturing. Leading vendors of automation equipment have

* Corresponding author. Tel.: +7-931-286-8401
E-mail address: sergei.s.zobnin@gmail.com

surpassed all imaginable expectations about development of complex robotic systems, means of information perception, increase in number of control channels, intellectualization of the low level of industrial automation such as PLC, fieldbus systems, sensors and actuators.

Knowledge usage in smart manufacturing can be divided into two main sections: 1) in domain for planning and execution domain-related actions 2) internally in manufacturing systems for optimization of manufacturing systems work i.e. independent actual application.

Multi-agent control is an important concept that wraps knowledge usage with additional benefits. It allows sufficiently extract functional abilities and efficiency of multi-purpose control in barely determined and a priory-undetermined exploitation conditions, which can be caused by complexity of controlled objects and existing of uncertainty of functioning in different undue, emergency and pre-emergency situations, when number of alternatives of scenarios is high enough and does not allow fast efficient decision making. Paper [1] may serve as an example of multi-agent framework that can be used to build a distributed control system. Authors of [2] developed a multitier control system with a self-organizing assembly subsystem at the bottom. Considered scenarios are similar to ones described in current paper with an exception of domain difference. Self-organization layer in [2] is used to robustly handle unexpected situations caused by both interior and exterior events (for example, “new assembly module” or “assembly module failure”) whereas in this paper self-organization is used to handle same type of events in information systems. While overall approach is similar in [1] and [2], authors of [2] provide engineering solution of a high-level planning problem. In [3] the work is extended by a cloud communication layer, which is similar to self-organization concepts described in current paper. Approaches with integrated intelligence for advance fault tolerance were summarized in [4], a technique presented in current paper may be considered as a particular case. Traditionally different techniques are used to improve execution speed of operations on time series. Examples of such specially developed techniques are [5, 6]. While authors focus on advanced dimensionality reduction and indexing techniques (that is, high-level operations on time series), current paper solves a more basic problem – more optimized approach of simple time series retrieval operations is defined.

For real usage theoretical concepts need to get a practical implementation with exploitation of modern engineering techniques. Modern engineering itself has recently started embracing theoretical concepts from distributed computing and applying it in designs. Some noticeable points from recent engineering are:

- Service decomposition – a way of splitting a monolithic piece of software on a logical and physical layer (separate deployment units) that provides better decoupling and higher flexibility
- Actively used replication and partitioning schemes, reduced consistency requirements
- Fail-fast designs instead of fail-safe designs, which are possible and should be embraced in case previous two points are considered

Modern design approaches allow facing challenges appropriately, but require additional engineering skills. There are also distribution-induced problems that are to be solved in distributed design: configuration, deployment, orchestration and logging. While special tools and frameworks were developed by community to help architects and programmers to deal with these problems, it may be arguable in some cases that solving those issues always takes less than profits gained by distribution and it would be definitely a win-win to have approaches that do not create additional problems.

Authors of this paper claim that system architectures, which are based on concepts of intelligent agents, provide benefits comparing to ad-hoc implementation of distributed systems and architectures. The main effects multi-agent technology based system can experience and exploit are:

- Presence of a decision making layer. Many everyday problems are a form of decision making. Distributed systems’ engineering can be simplified by using intelligence rather than ad-hoc distribution and coordination techniques
- Theory-based fault tolerance and scalability. Advanced distributed coordination and decision making techniques allow more effective operation and availability
- Presence of internal general computational framework that can be used to solve business tasks. For example, same optimization routines being a part of underlying multi-agent framework can be used for solving data mining problems

The rest of this paper is organized as follows. Sections 2, 3, 4 present different aspects of multi-agent approach implementation for smart manufacturing applications. Presented techniques are aimed on different system characteristics improvement – scalability, fault-tolerance and inter-system latencies. Section 5 presents results of modeling and prototypes, interpretation of those results and further work. Section 6 describes a laboratory prototype where the described approaches are validated before production usage.

2. Hybrid P2P-backed multi-agent topologies for better scalability and fault-tolerance

A novel architecture for distributed automation systems aiming on efficiency, high levels of scalability, high levels of fault tolerance and featuring capabilities of workload distribution between traditional server agents and thin client browser-based agents in environments with continuously arriving data streams is presented in the following paragraphs.

Authors of [7] claim that a web browser is a very convenient environment for multi-agent system execution. Recently rich web-based gained popularity due to the absence of installation procedures, which is very convenient for the end user. The same benefit can be used in a general multi-agent system: instead of administration complexity - easy deployment scheme can be considered: in order to extend (scale) a multi-agent system end user needs to open an additional web browser (page can also be opened programmatically). There are also separate web browser engines, can be opened without interface as a background process. Modern standards and APIs [8, 9] make it possible to implement functionalities that are currently considered only as server-side ones. Using web browser as an engine for multi-agent system also exploits existing users' performance in order to optimize server load – currently all modern systems are web-based.

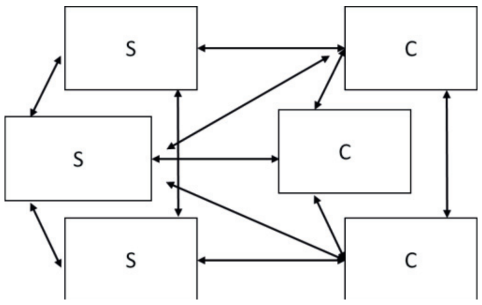

Fig. 1. Hybrid multi-agent system topology.

A hybrid approach – when there are both standard server agents and client agents – is presented in Figure 1. When using both types of agents, though, some problems due arise – standard mathematical frameworks are developed for single type of agents, but this is a minor issue. When considering online data, the problem of channel source selection arises, which is both a problem of optimal fault-tolerance and load-balancing schemes.

A channel source selection problem for hybrid P2P-backed multi-agent system applied in production systems is presented in following paragraphs.

Let us consider a problem of online data transmission to a set of client agents from a set of server agents. As stated earlier, important benefits of decentralized control, organization and execution include scalability and fault-tolerance. Providing these benefits in distributed environments requires solving an optimization task in a runtime – a client agent should establish, maintain and support a best subset of possible connections with other client and server agents.

The following assumptions are made among obvious in channel source selection problem definition:

- Client agents require physical bandwidth for communication for each of connections, i.e. no VM sharing is assumed
- Each server agent spends server time for client agents' channels serving. Servers' time is more important than clients', hence server load should be minimized as a part of problem solving

- Network interruptions are possible, a client should be able to receive channels' data with high likelihood in online fashion and take into account dynamically changing environment
- A client requires data in online fashion meaning that optimization should be performed also in online fashion (continuously) and there is a time constraint for optimization (one second delays are considered)
- It is assumed, that when a client agent is created, some server agents are already launched
- Server agents contain all channels' data in each moment of time

In this section notation is introduced for channel source selection problem in hybrid P2P architecture that backs a multi-agent system described previously.

Let c_i $(i = 1 \dots n)$ denote a channel of the automation system. Let n_j $(j = 1 \dots m)$ denote a client agent. Let s_k $(k = 1 \dots q)$ denote a server agent. Let e_t denote an epoch which itself stands for a particular moment in environment's time line. In each epoch e_t an optimal solution is found for e_{t+1}, a one step look ahead.

Let $conf_{c_i,n_j}$ and $conf_{c_i,s_k}$ denote a confidence coefficient which itself denotes a likelihood of piece i presence in n_j and s_k respectively. Confidence levels (coefficients) are maintained internally by each client agent and depend on epoch (i.e. $conf_{e_t,c_i,s_k} != conf_{e_{t+1},c_i,s_k}$), but due to the fact only next epoch is considered, epoch indices are omitted in the notation.

Let $conf\_adm_i$ denote a minimal admissible confidence of receiving a channel i in the following epoch e_{t+1}. An admissible confidence does not depend on epoch (i.e. $conf\_adm_{c_i} = conf\_adm_{e_t,c_i} = conf\_admissible_{e_{t+1},c_i}$).

Let x_{c_i,n_j} and y_{c_i,s_k} be binary variables that denote the fact than in the following epoch current client agent is downloading channel c_i from neighboring client agent n_j and server s_k respectively.

A client agent maintains a set of opened connections with a subset of other client agents and a subset of server agents, following summations are performed over client and server agents, for which connections are established by the current agent (active subset of client and server agents).

In each epoch, a client agent maintains a sufficient confidence of receiving all required channels in the following epoch. The constraint (1) is maintained for each channel.

$$\sum y_{c_i,s_k} * conf_{c_i,s_k} + \sum_{n_i} x_{c_i,n_j} * conf_{c_i,s_k} \leq conf\_adm_{c_i} \tag{1}$$

Let b be a bandwidth overhead for a connection between two client agents. A client agent is trying to minimize total bandwidth overhead (2) for each channel. For simplicity, bandwidth overhead is constant.

$$\sum x_{c_i,n_j} * b \to min \tag{2}$$

Let l_{s_k} denote current load of server s_k. A client agent maintains a server loads vector updated in real time for each server agent it is connected to. From local perspective, a client agent minimizes a total server load (3) for each channel that has active connections to server agents.

$$\sum_{s_k} y_{c_i,s_k} * \ l_{s_k} \to min \tag{3}$$

The multi-criteria optimization problem is defined as follows:
Minimize

$$\alpha \sum_{c_i} \sum_{n_i} (x_{c_i,n_j} * b) + \beta \sum_{c_i} \sum_{s_k} (y_{c_i,s_k} * \ l_{s_k}) \to min \tag{4}$$

where α and β are coefficients

Subject to

$$\forall c_i \; in \; active \; subset:$$
$$\sum_{s_k} y_{c_i,s_k} * conf_{c_i,s_k} + \sum_{n_j} x_{c_i,n_j} * conf_{c_i,s_k} \le conf_{adm_{c_i}} \tag{5}$$

$$y_{c_i,s_k} \in \{0,1\}$$
$$x_{c_i,n_j} \in \{0,1\}$$

A client agent lifecycle algorithm is described in the following paragraphs. Textual description follows actual algorithm depicted in Fig.2.

```
Algorithm 1: Client agent lifecycle for epoch e
  Input: $initReqChannels$, $initClientConn$,
         $initServerConn$
1  $reqChannels$ = $initReqChannels$
2  $clientConn$ = $initClientConn$
3  $serverConn$ = $initServerConn$
4  $clientWeights$ = initClientWeights()
5  $serverWeights$ = initServerWeights()
6  while executing do
7      if nextEpochCame then
8          $serverLoads$ = receiveLoads()
9          $reqChannels$ = updateRequests()
10         $clientConn$ = updateClientConns()
11         $serverConn$ = updateServerConns()
12         $clientWeights$ = updateClientWts()
13         $serverWeights$ = updateServerWts()
14         ($clientConn$, $serverConn$) = optimize()
```

Fig. 2 Client agent lifecycle algorithm.

When client is initialized, it has internally a list of required channels. If a client agent is a backed off a web-based interface, requested channels list is simply list of channels requested by the end user. A client agent chooses randomly initial server and client agent to be connected to. An exact protocol is not aimed to be defined, that may be either a local algorithm or a client-server protocol. Initial confidence coefficients are set up (lines 4 and 5), for server agents confidence levels are high because of assumption. Client agents' confidence levels are low, because initially there is no information on other clients' requested channels. Then an execution loop is started.

A client agent receives server loads on every epoch (line 8) – this ensures that servers are properly utilized. On each iteration client updates if necessary client and server connections in order to handle external uncontrolled changes (lines 10 and 11). An internal history of absence and presence of data on remote client and server agents is tracked. On lines 12 and 13 a client agent performs update of these weights before each optimization step. Finally, on line 14, optimization routine is called.

3. Integrated intelligence for advanced fault-tolerance

For systems build as a union of separate agents one may design a layer of integrated self-supervision and control powered by internal intelligence [10]. A view of decision-making process is presented for a single agent in the following paragraphs.

Partially observable Markov decision processes (POMDP) model was chosen as a basis of integrated decision making subsystem. POMDPs are a proven technique of making value-oriented decisions in presence of observation and action uncertainty. POMDP is defined as a tuple M = (S, A, O, T, Ω, R), where

- S is a set of states
- A is a set of actions
- is a set of observations
- T is a set of conditional transition probabilities
- Ω is a set of conditional observation probabilities
- $R : A \to \mathbb{R}$ is the reward function

The example of system architecture with POMDP solver is showed in Fig.3:

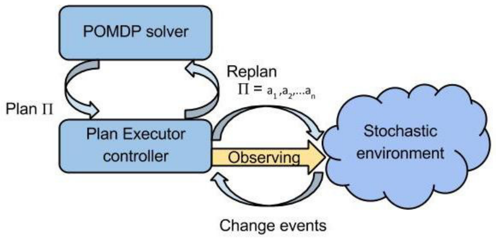

Fig. 3. Fault tolerant component architecture.

A basic set of possible actions can consist of the following values:

- “Do nothing”
- “Reload component”
- “Turn component on”
- “Turn component off”

Decision making system/layer reads status messages locally in each component and performs local supervisory actions. Status messages are inherently observations in POMDP model. Internal system state is identified in the process of arriving observations. Considered component states are the following:

- Down – component is down and can be reinitialized by means of local supervision/orchestration
- Crushed – component cannot be reinitialized without human interaction
- Incapable – component is still alive but cannot perform according to specifications
- Working – component functions according to specifications

The following paragraphs describe how POMDP can handle component state evolution to improve system fault-tolerance.

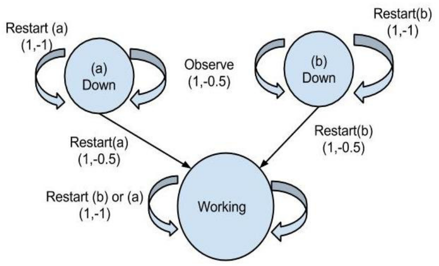

Fig. 4. Recovery model.

Application of POMDPs for the recovery process of two components (a) and (b) is depicted in Figure 4. Component can observe it's state and take appropriate value-oriented decisions.

POMDP policy is a mapping between state and action for each initial state of the system. The optimal policy, denoted by p*, yields the highest expected reward value for each belief state, compactly represented by the optimal value function V*. This value function is solution to the Bellman optimality equation:

$$V^*(b) = \max_{a}[r(b,a) + \gamma \sum \Omega(o|b,a)V^*(\tau(b,a,o))] \tag{6}$$

where γ is the discount factor, τ is the belief state transition function and b is a belief state probability. The value function can be found with a variety of algorithms. Partial observability makes the problem hard to solve in a naïve way, different approaches were developed to tackle complexity: online approaches, approximate offline approaches, hybrid approaches [11, 12, 13].

4. Advanced adaptive storage scheme for low-latency analytics and visualization

Storage subsystem for time series data is important aspect of modern automation systems. A novel adaptive approach of storing was developed by authors of [14] and is presented in this chapter. Realization of approach is based on modern NoSQL technologies (for references see [14]). Approach is based on utilization of user behavior patterns.

The key problem of optimal storage in NoSQL storage is identification of a proper source channels separation. If an information system produces thousands values per second, one may store all of them altogether, store each value separately or chose an intermediate scheme. Here and further block is a subset of channels that are subjected to be stored in one storage unit.

The idea behind finding optimal blocks is to introduce heuristic function and minimize it. Optimal data structure is the result of this operation. This structure can be used to divide original object to blocks. This process can be repeated after arbitrary time.

Let EQT denote expected query time. Let R denote retrieved from database binary data. Let T denote elapsed time to retrieve data from block (deserializing time). Let s denote schema – a way for dividing data to blocks. Let d_i ($i = 1..n$) denote one of user requests, channels that were requested. Heuristic (7) after minimization provides an optimal (in sense of request time / system latencies) structure:

$$E\mathrm{QT}(\mathrm{s},\mathrm{d}) = \sum_{i} T(R(s,d_i)) \tag{7}$$

Heuristic (7) exhibits a sufficient drawback, it prefers solutions which follow users' patterns, which can be considered as a generalization issue. Heuristic (8) is an improved version, which is less greedy and provides better solutions.

Let ESA represent an expected storage amount, technology-specific value.

$$ESA = \sum_{sensorId} \sum_{blockid} count(sensorId)_{blockid} \tag{8}$$

Minimizing F (9) we achieve optimal structure with better generalization characteristics.

$$F = E\mathrm{QT} + \beta \cdot \mathrm{ESA} \tag{9}$$

Parameter β is responsible for controlling data overhead. Picking β = 1 will not permit data overhead and increase latencies of resulting scheme and vice versa.

5. Results and further work

Applicability of three described techniques was investigated with modeling techniques and prototypical implementations. Figure 5 shows modeling results of heterogeneous connectivity as described in section 2.

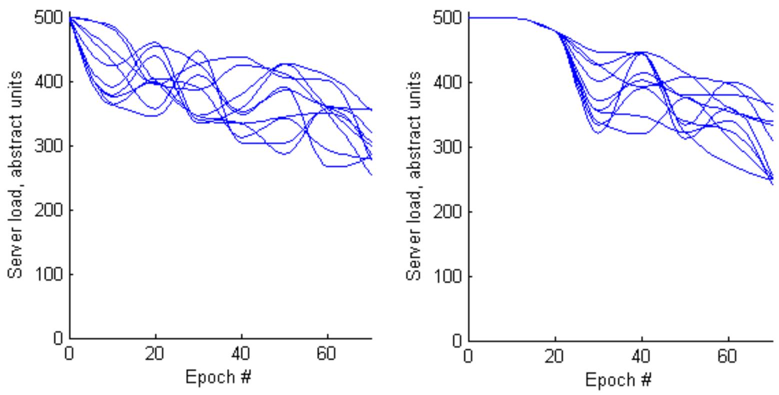

Fig. 5. (a,b) Results of modelling server loads for different starting conditions.

Figure 5,b describes a system behavior in case starting client weights are low. Client agents collect confidence and then start distributing. Figure 5,a describes a default case where client agents have random connections with nonzero confidence values. Oscillations occur due to random events that were generated in environment modelling algorithm. Ongoing work is generation of more advanced models of client / server agents behavior and failure events. In parallel a real prototype of such connectivity is tested and results are collected to verify and improve models. Developed framework of hybrid peer-to-peer connectivity between client and server agents is going to be open sourced. Preliminary results of prototypes look promising and alike first modeling results.

While small-scale modeling of POMDP fault handling has proven it's applicability, real-world problem size may become an issue. A big POMDP for is being developed, as well as distributed algorithm to solve the defined model. In this model actions and observations are extracted from logs of real distributed systems.

Adaptive storage scheme looks promising for cases of stationary channel sets. Further development is an adoption of technique to cases where channel sets are changing over the time. The technique minimized latencies and maximized performance in a production time-series based system.

Results of modeling and prototypical implementations are consistent in their success with analogous techniques in other domains. Information systems domain provides even better starting conditions due to absence of technological problems arising in wider domains.

6. Concepts implementation

Concepts described in previous sections are used as basic building blocks of automation systems that are designed and built on the premises of Festo Laboratory in Saint Petersburg Polytechnic University. The lab consists of two manufacturing models – a discrete and continuous manufacturing.

Fig. 6. Festo Laboratories in SPBSPU.

Among a variety of projects, two are the most noticeable due to their completeness in terms of smart manufacturing ideas implementation. The first one, “Control of Continuous Technological Dynamic Processes”, features a distributed decision making expert system for processing of scenarios for warning and prediction of out-of-order emergency situations. The second one, “Decentralized Control Network in Distributed Systems and Technological Processes”, features a multi-agent optimum distribution - a platform of computing resources of control network for the coordination of diverse subsystems, elements and communications.

7. Conclusions

The beautiful Smart Manufacturing concepts require a lot of engineering and scientific problems to be solved and implemented in order to become real. This work has presented several distinct techniques that are being researched and implemented while being parts of customer and research projects. Described techniques improve characteristics of resulting automated systems comparing to traditional approaches of design and implementation: fault-tolerance, scalability and low latencies; that fact is proved by systems designed and implemented on the premises of Festo Laboratory in Saint Petersburg Polytechnic University and in customer projects.

References

[1] D.G. Arseniev, V.P. Shkodyrev, V.V. Potekhin, V.E. Kovalevsky, Smart Manufacturing with Distributed Knowledge-Base Control Networks., in Proceedings of Symposium on Automated Systems and Technologies, Hannover, PZH Verlag, 2014, pp. 85-89.

[2] B. Katalinic, V.E. Pryanichnikov, K. Ueda, Kukushkin, I., P. Cesarec, R. Kettler, Bionic assembly system: hybrid control structure, working scenario and scheduling, in Proceedings of 9th National Congress on Theoretical & Applied Mechanics, Brussels, 2012, pp. 111-118.

[3] B. Katalinic, I. Kukushkin, V. Pryanichnikov, D. Haskovic, Cloud Communication Concept for Bionic Assembly System, Procedia Engineering, Volume 69, 2014, Pages 1562-1568, ISSN 1877-7058, doi:10.1016/j.proeng.2014.03.156.

[4] B. Lussier, R. Chatila, F. Ingrand, M. O. Killijian, D. Powell, On fault tolerance and robustness in autonomous systems, In Proceedings of the 3rd IARP-IEEE/RAS-EURON Joint Workshop on Technical Challenges for Dependable Robots in Human Environments, 2004.

[5] E. Keogh, K. Chakrabarti, M. Pazzani, S. Mehrotra, Locally adaptive dimensionality reduction for indexing large time series databases, ACM SIGMOD Record, 30(2), 151-162.

[6] E. Keogh, K. Chakrabarti, M. Pazzani, S. Mehrotra, Dimensionality reduction for fast similarity search in large time series databases. Knowledge and information Systems, 3(3), 263-286.

[7] S. S. Zobnin, V. V. Potekhin, P2P architectures in distributed automation systems, in Proceedings of Symposium on Automated Systems and Technologies, Hannover, PZH Verlag, 2014, pp. 37-43.

[8] Web Workers (2012). Retrieved July 10, 2014 from http://www.w3.org/TR/workers/.

[9] WebRTC 1.0: Real-time Communication Between Browsers (2014), Retrieved July 10, 2014 from http://dev.w3.org/2011/webrtc/editor/webrtc.html.

[10] A.V. Fedorov, S.S. Zobnin, V.V. Potekhin, Prescriptive analytics in distributed automation systems, in Proceedings of Symposium on Automated Systems and Technologies, Hannover, PZH Verlag, 2014, pp. 43-49.

[11] S. Ross, J. Pineau, S. Paquet, B. Chaib-Draa, Online Planning Algorithms for POMDPs, J. Artif. Intell. Res.(JAIR), 32 (2008), 663-704.

[12] J. Pineau, G. Gordon, S. Thrun, Point-based value iteration: An anytime algorithm for POMDPs. In IJCAI (Vol. 3, pp. 1025-1032).
[13] M. T. Spaan, N. A. Vlassis, Perseus: Randomized point-based value iteration for POMDPs. J. Artif. Intell. Res.(JAIR), 24 (2005), 195-220.
[14] E.S. Goloshchapov, S.S. Zobnin, V.V. Potekhin, Adaptive Time Series Storage Scheme, in Proceedings of Symposium on Automated Systems and Technologies, Hannover, PZH Verlag, 2014, pp. 115-119.