Multi-agent Autonomous Group Control in Collective Robotics-Based Assembly

Vladimir Gorodetsky[a]

[a]St. Petersburg State Electro-Technical University (LETI) after V.I. Ulyanov (Lenin), Saint-Petersburg, Russia

Abstract

The paper scope is autonomous operational group control over the assembly of a complex product in manufacturing performed by a team of interacting autonomous robots. Its contribution is agent-based model of robots' individual behavior formalized in terms of finite state machines with internal states and distributed self-organizing algorithm (protocol) coordinating in peer-to-peer mode goal-oriented collective behavior of robots' assembly team operating without any external intervention. An example illustrates the paper contribution.

1. Introduction

During the last decade, one can observe an ever-increasing interest of the research and industry communities to autonomous collective robotics. Gradually, this interest is actually transforming to a long-term trend. The potential application areas of collective robotics is wide enough, e.g. building, precise agriculture, space and underwater applications, perimeter security systems, collaborative load transportation, among others. However, manufacturing remains to be the first class of collective robotics applications. In it, automatic and autonomous assembly of complex products like airplanes, cars, etc., with an emphasis on the product customization is nowadays paid the special attention. Advantages of such assembly technology are manifold and well recognized. However, a barrier to the wide use of autonomous and automatic assembly exploiting collective robotics is under-development of efficient and effective theoretical and algorithmic basis in the scope of group behavior control in real-time mode.

The paper studies the aforementioned problem with the accent on the fully autonomy of the teamwork of assembly robot performance excluding any external intervention if the assembly scenario is being performed well, correctly and timely. It investigates the problem from theoretical point of view and takes into account the application-oriented context determined by the specificity of the collective assembly in manufacturing. The model of autonomous agent is applied to conceptualize assembly robot's individual behavior that is further formalized in terms of finite state machine with internal states. This formal model is enough expressive to specify proactive individual behavior of robot coordinated according to the group scenario. Accordingly, the collective behavior of assembly robots' team is represented as a multi-agent system

Russian Advances in Artificial Intelligence: selected contributions to the Russian Conference on Artificial intelligence (RCAI 2020), October 10-16, 2020, Moscow, Russia

vladim.gorodetsky@gmail.com (V. Gorodetsky)

(MAS), constituting the network of interacting autonomous agents. The formal specification of MAS architecture is given in terms of network of state machines exchanging messages in order to coordinate the agents' individual behaviors in distributed fashion.

Thus, the paper objective is to present the developed solutions comprising the state-machine-based formal models of agents and distributed algorithm (protocol) implementing coordination of individual robots' behaviors according to the particular scenario implementing the goal-oriented group behavior.

In the rest of the paper, section 2 reminds the basic properties of autonomous agents and MAS that are of great importance for the control model of group behavior. Section 3 describes the contents and senses of the basic concepts of group behavior model built in the paper. Section 4 introduces the developed architecture of MAS comprising the models of the individual behaviors of assembly robots, the structure of information environment, in which they operate, and their interactions implementing the distributed coordination of individual robots' behaviors to achieve the group goal. This section specifies the roles and functionalities of agents' classes introduced too. Section 5 specifies the formal models of agents' individual behaviors in terms of interacting state machines with internal states and distributed algorithm (protocol) of self-organizing coordination of agents' individual behaviors implementing the group control. Section 6 outlines an example illustrating the basic components of the proposed MAS-model and basic ides of autonomous group control of robots' individual behaviors coordinated in distributed self-organizing fashion according to the group behavior scenario. Section 7 outlines related works. Conclusion summarizes the paper contribution and outlines the perspective of the future research in the context of the paper.

2. Key Properties of Autonomous Agents and MAS

Software agent is an autonomous software program (system) that exhibits a goal-directed proactive behavior in dynamic unpredictable environment without external intervention. Its basic properties are autonomy, the capability to exhibit persistent goal-directed behavior and control own internal states. Let us explain these properties.

Autonomy is the ability of agent to preserve persistent goal-oriented behavior in dynamic environment without external intervention, e.g. without intervention of a user. In other words, when autonomous agent finds out in a new situation inspired e.g. by dynamics of environment it is nevertheless is capable to compute new behavior resulting in achievement of the same goal. Other, more practice-oriented definition was given in [1]: "Autonomy is the ability for an agent to solve local and individual problems using its own memory and ability to decide what to do next in accordance with what kind of information it perceives".

The ability of agent to exhibit a *proactive behavior* means that the agent, to make a decision, takes into account not only information perceived from the external environment, but also prehistory of the external environment states and history of its own behavior reflected somehow in its current internal state. For instance, the agent can generate events and send messages to other agents of MAS even if it gets no input information. These proactive messages can be inspired, e.g., by disruption of a timeout measured by the agent itself. It is worth to note that proactivity is the main distinctive feature of an autonomous agent in contrast with an object,

as the latter is understood in Object-Oriented Programming (OOP) [2].

Multi-agent system (MAS) is a network of weakly coupled software agents solving particular problems, situated in common environment, and interacting with each other to cooperate and/or to coordinate their behavior in order to achieve their common goal or their own particular ones.

Interaction is the third key feature of a software agent in MAS, in addition to autonomy and proactivity. Interactivity is the agents' ability to influence each other in a way and exactly the latter determines the sense of the term "weakly-coupled".

"From interaction and autonomy comes … *emergence*, i.e. the ability to produce new results and solve complex problems as a side effect of all the particular actions that agents perform through their coordination of action with other agents." [1]. Thus, the emergence is one more feature of MAS that one has to take into account in agent-based group behavior modeling, and the emergence is the basis of self-organization.

3. The Fundamental Concepts of Group Behavior Modeling

Although the group control-related researches have more than thirty years history, this problem still has no semantically consistent conceptual basis adopted by all. However, the existence of such basis is highly needed from different points of views and the first of them, among others, is specificities of the many modern applications. Indeed, as a rule, these applications comprises formidable number of distributed heterogeneous components operating in cross-domain semantic space and, therefore, to coordinate their individual behaviors towards the common goal through message exchange, they need shared ontology-based conceptual basis.

In this respect, [3], [4], [5], [6], [7], [8] proposed various versions of ontology models for group behavior and group control problem as applied to the collective robotics. However, [3], [4], [8] proposed the ontologies dealing with, to a greater degree, concepts peculiar to individual robots' behavior, whereas [5], [6], [7], [8] proposed some variants of domain-oriented conceptual bases.

Although the development of ontology for group control problem is not the subject of the paper too, it is necessary to refine what the concepts used in the paper mean. These refinements are given below.

Group is two or more autonomous entities (physical, social or virtual) depending on each other in some context, e.g. they participate in solution of shared task, they compete for shared resources, etc.

Individual goal of an autonomous entity (e.g., robot) in a group behavior scenario is a set or a sequence of states the entity has to achieve according to its commitment in this scenario.

Individual behavior of an autonomous entity (robot, for instance) is a sequence of tasks it has to solve as a member of the group according to its commitments.

Group goal is a structured set of states that the group has to achieve, and, at that, any individual goal of a group member is a component of the group goal tree, and, thus, it is a mandatory part of the common goal of the group as a whole.

Group behavior is a structured set of individual behaviors of the autonomous entities (robots) of the group coordinated to achieve the group goal.

Group control is a set of particular tasks designed to control over the individual behaviors and individual goals of the group members intended to achieve the group goal.

Team is a group of autonomous entities (robots), which individual goals and behaviors fully subjected to the achievement of the common group goal.

Autonomous control over group behavior is an option of the group control if the group goal and, perhaps, operation scenario designed to achieve it is designated by an external entity, whereas allocation of the scenario tasks over the group members (robots) and real-time control of the scenario execution is solved by the means of the group itself.

Scenario of actions is a partially ordered set of actions the group has to perform in order to achieve the group goal.

Scenario of a group behavior is a scenario of actions together with the autonomous entities (robots) allocated to perform these actions.

Group commitments of a group member is the list of the tasks it has (intends) to perform according to the scenario of the group behavior.

Group conventions of a group member is the list of conditions determining when all the group members have to break their commitments.

Joint Intention Protocol [9] is an auxiliary distributed algorithm the group members have to implement before starting the group behavior. This protocol is applied to get and to check the confirmations from all the group members that they are informed about and intend to fully implement their commitments and conventions as a part of the scenario of group behavior, i.e.

1. to follow the allocated commitments, and
2. to break the scenario performance if at least one condition of the conventions is met and to undertake the steps to inform the group members about this fact.

Individual commitments and conventions is a part of group commitments and group conventions allocated to the particular group member.

Situation is a dynamic characteristic of a system representing what was and is going in it projected for the future [10], [11], [12].

Situational awareness of a group member is its state of awareness in which it holds the sufficient volume of information to perform own group commitments and conventions according to the scenario of a group behavior and to synchronize and coordinate own individual behavior accordingly [13]. Most of the aforementioned concepts are illustrated by examples below.

4. Scenario of Actions, Scenario of Group Behavior, Situation, Expanded Scenario Graph

Group control task comprises two typical subtasks:

1. Allocation of the structured set of scenario actions implementing goal-oriented behavior of the group constrained by the robots abilities (“action planning”), and
2. Operational management (control) intended to coordinate and synchronize the group behavior and mitigate the scenario performance deviations.

The first subtask is domain-oriented. It has many statements constituting a specific planning problem group control. However, this task is out of the paper scope. In the illustration example (section 7), one of such planning algorithm of heuristic nature is implemented. In contrast, the second task is domain independent and it is one of the two main paper subjects.

Figure 1 illustrates graphically an example of a scenario of actions to be performed by a robot team. This scenario represents a partially ordered set of particular actions specified formally as given below:

$$\begin{aligned} \boldsymbol{AX} = \{\boldsymbol{X}, >\} = \{\boldsymbol{X}, \{(X_4 > X_1), (X_3 > X_1), (X_3 > X_2), (X_5 > X_2), (X_6 > X_2), (X_7 > X_4), \\ (X_8 > X_3), (X_9 > X_3), (X_{10} > X_5), (X_{10} > X_6), (X_{12} > X_9), (X_{11} > X_{10}), (X_{12} > X_{11})\}\} \end{aligned} \tag{1}$$

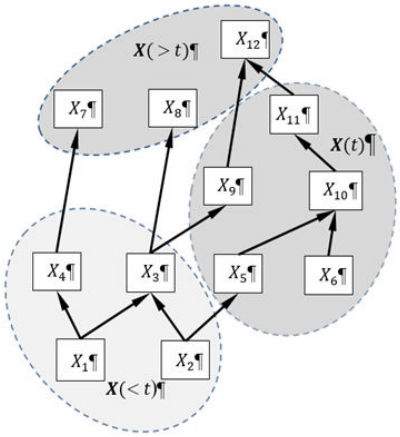

Figure 1: Scenario action graph partitioned in three subsets: $X(< t), X(t)$ and $X(> t)$

In its graph-based representation, nodes represent the scenario actions and arrows – partial order relations, at that the destination node of an arrow indicates the immediately later action for its source node. In fact, action scenario is a knowledge structure specifying the set of actions and their ordering.

This graph represents the whole scenario of actions, i.e. the state of its execution at the start time. In progress of the scenario performance at the time instant t, all the scenario actions can be factorized into the subsets of three categories:

- accomplished actions, denote this subset as $X(< t)$;
- actions in progress, denote this subset as $X(t)$;
- still not started actions, denote the corresponding subset as $X(> t)$.

An example of such partition is given in Figure 1.

This partition provides valuable information used for operational control over the scenario action performance. However, the scenario action graph does not represent this information

and, thus, it is unable to represent the state of the action scenario performance at arbitrary time moment t, in particular, to represent real-time dynamics of the partition introduced in Figure 1. To specify formally this information, let us define a data structure called *expanded scenario graph* (ESG) in the way described below.

For each node (action) X_j of the scenario graph, we introduce a new node denoted F_k that immediately follows X_j. In this structure, function of the node F_k is to represent the *Status of the immediately Preceding Action* (SPA) and make this information available to all its immediately successive nodes (actions), in the scenario. Thus, the set of "fired" F_k indicates the boundary between the subset of completed actions $X(< t)$ and the other ones. Additionally, the expanded scenario graph definitely determines the sunset of actions that are ready for performance. The new nodes inserted into the action scenario graph in this way are called below *SPA-nodes*, for short.

As usual, let us also introduce, in the scenario action graph, two fictitious nodes:

- *starting node* X_S preceding all the scenario action graph nodes with no *input* edges,
- *final node* X_F following all the scenario action graph *nodes* that have *no followers*.

The starting node X_S is also assigned the SPA-node F_S following the former. Figure 2 explains how ESG is built. Figure 3 presents the expanded scenario graph built via transformation of the action scenario (Figure 1) according to the rules shown in Figure 2.

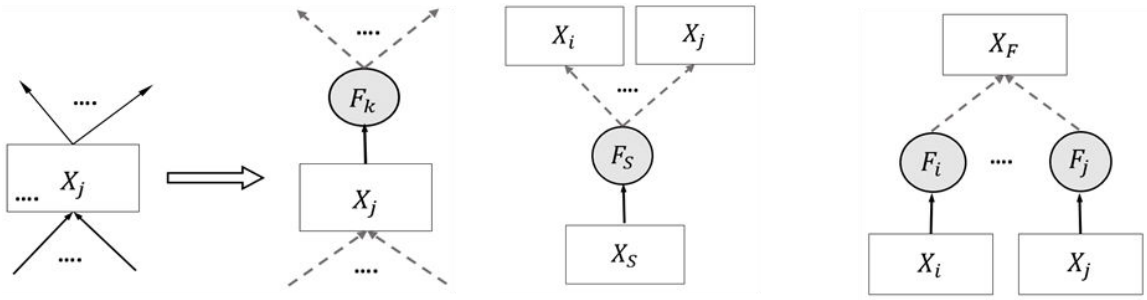

Figure 2: The rules to transform an action scenario to the expanded scenario graph: a) arbitrary node X_j;
b) initial fictitious node X_S; c) final fictitious node X_F.

An important concept of the group control is *situation* outlined in section 3. This concept has been introduced in [10], [11], [12] as applied to group behavior systems; its role in a group behavior ontology has been summarized in [6]. According to its definition, the situation is specified in terms of the states of the objects comprising the system and what is more important in terms of relations between the system objects:

$$\boldsymbol{S} =< \{Y_k \mid_{k=1}^{N}\}, \{E_r \mid_{r=1}^{M}\}, t \in [t_0, t_k] >, \tag{2}$$

where $Y_k \mid_{k=1}^{N}$ is the set of states of the system objects (e.g. robots and nodes of the expanded scenario graph, for application in question), $E_r \mid_{r=1}^{M}$ is the set of relations given over the system objects.

One can assess a situation from multiple viewpoints, and these assessments are usually measured in qualitative scales using various classifications (factorizations) of the situation set that can happen in the system. For example, from the status of goal achievement viewpoint, the set of situations can be factorized in four subsets: {*goal is achievable, unachievable, is achieved, irrelevant*}. E.g. for the product assembly, goal is *unachievable* if there is no robot with a required capability. The goal is *rrelevant* if the assembly process is broken according to the outside event, for instance. One more example of situation assessment point of view is its performance that can take values from the following set of situation classes: {*normal, exceptional, emergent*}.

5. Multi-agent Architecture and Self organizing Group Control

The developed MAS-architecture of autonomous control of teamwork of robots performing assembly of a complex product contains the agents of various classes.

Agent of robot is the software agent class mapped to every assembly robot participating in the group assembly process. Its functionality is to control over the robot individual behavior according to its particular commitments and shared conventions and to coordinate this behavior with individual behaviors of other robots allocated their own commitments and conventions according to the Joint Intention Protocol too [9].

Two classes of agents are introduced to support for the distributed situational awareness of particular robots needed to coordinate their individual behavior according to the online state of the performance of the action scenario. These agent classes are assigned to the nodes of the ESG-data structure (see an example in Figure 3) to represent real-time evolution of the latter. The first of these two agent classes is *Action agent* class and the second one is *SPA-agent* class with the instances assigned to every node of the ESG-data structure. In the software agent world, the instances of these agent classes represent the online statuses of *Action-nodes* and *SPA-nodes* of ESG, respectively.

An additional auxiliary agent role that can be assigned either to any other agent instance or to a specific software agent instance is the role of *Leader*. Its functionalities are on-line system behavior monitoring and initiating some group activities of robots, e.g. re-planning if the group behavior deviations is significant and needs corrections.

Thus, the proposed MAS-architecture of real-time distributed control system of a robot team performing autonomous assembly of a product is represented in terms of a *network* of interacting *autonomous software agents*. In this network of agents, the assembly robots are active entities playing the role of effectors whereas *Action agent* and *SPA-agent* instances produce on-line information required to support situational awareness of the active entities about on-line state of the action scenario performance.

Implementation of the distributed situational awareness (see Section 2) of robots is the first class task, for coordination of the robot individual behaviors. It is implemented through the following agent information exchange strategy:

1. Instance of *Agent of action* X_j monitors the states of its predecessors in the expanded scenario graph to detect the fact that all its predecessors have been transited from the set of action in *progress* $X(t)$ into the accomplished action set $X(< t)$ (Figure 1).

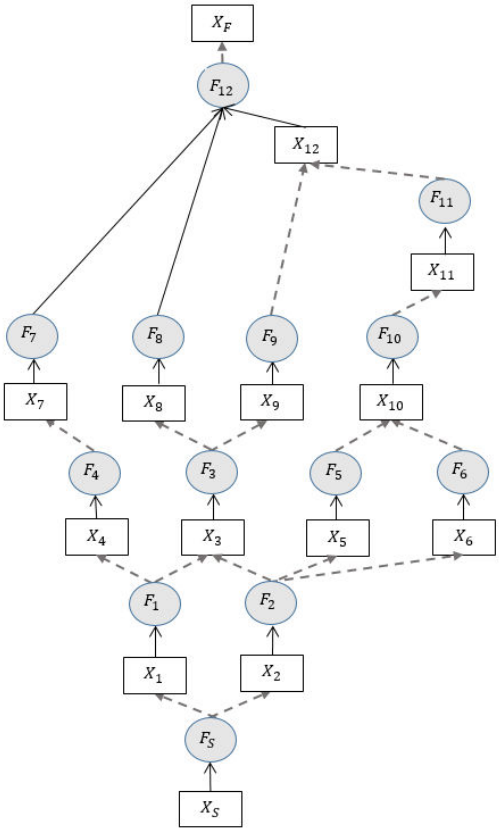

Figure 3: Expanded scenario graph – a data structure specifying dynamics of the action scenario performance

2. Instance of *Agent of action* X_j "knows" allocated robot and sends message to it when the aforementioned event is detected.
3. Instance of *Agent of robot* R_i allocated to perform the action X_j has the latter in its ordered list of commitments, and when it has accomplished the previous action of this list it becomes ready to perform action X_j. Next, either robot R_i waits readiness message from *Agent of action* X_j or the latter is waiting for the release of robot R_i.

This information exchange supports for situational awareness of robots required to them to coordinate their individual behaviors according to the action scenario and robots' commitments. The main advantage of this strategy is that situational awareness is achieved on the basis of local interactions of agents of robots, from the one side, and agents of actions, from the other one thus implementing on-line self-organizing control.

6. Formal Model of Coordination of Robot Group Behavior

To specify the agents' individual behaviors, the formal framework of Finite State Machines (FSM) with internal states is selected.

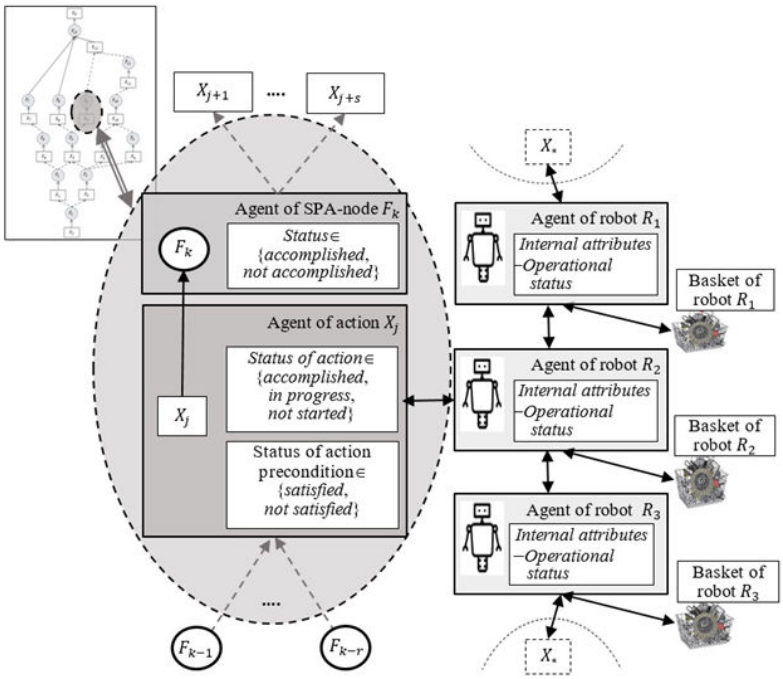

Figure 4: Local formal model of autonomous group control of robotic assembly process: agents, state machines, attributes of state machines, and their interactions

The instances of *Agent of robot* class are specified as FSM with one attribute of internal state (Figure 4):

- Operational status of robot - it is also two-value attribute taking value from the set {*available, not available*}.

The instances of the *Agent of SPA-node* class (e.g., F_k-node in Figure 3) are formalized in terms of FSM having single attribute *Status of preceding action* X_j taking value from the set {*accomplished, not accomplished*} depending on the status of X_j (Figure 4).

The instances of *Agent of action* X_j class are formalized in terms of FSM having two attributes of internal state (Figure 4):

- *Status of precondition of Action* X_j taking value from two-value set {*satisfied, not satisfied*}; it takes the first value if all the action preceding X_j are accomplished;
- *Status of Action* X_j taking value from the set {*not started, in progress, accomplished*}.

An example of MAS-architecture of group control system for autonomous robotic assembly process is depicted in Figure 4, where the formal models of individual behaviors of locally interacting agents are presented too. Let us emphasize that, in this architecture, local interactions of agents are restricted inside a group of agents comprising *Agent of action* X_j , *Agents* of its immediately preceding and immediately successive *SPA–nodes* and *Agent of robot* R_i that is responsible to perform the action X_j , according to the robot commitments. Figure 4 explicitly shows that the data structure representing ESG of the action scenario can be interpreted as a blackboard supplying the robots with the information needed them to coordinate scenario action performance, i.e. to provide for situational awareness.

Coordination and synchronization of the aforementioned local group of agents' behaviors is fulfilled according to the Group control protocol depicted as UML-like sequence diagram in Figure 5. Let us explain the message exchange assumed in it.

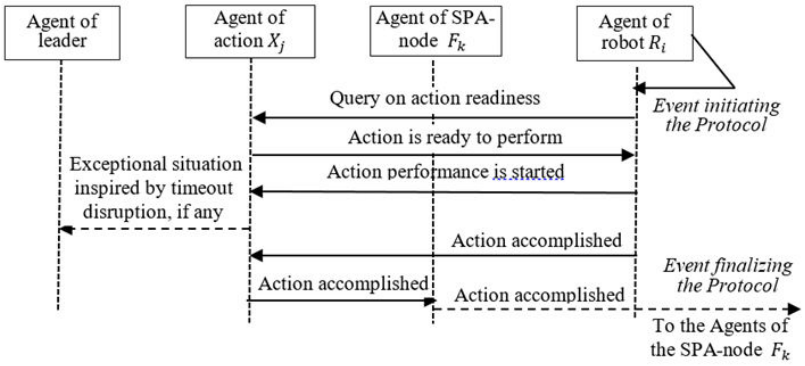

Figure 5: Sequence diagram of the group control Protocol – interaction of the related agents

The protocol is initiated by the *Agent of robot* R_i responsible for performance of the action X_j , according to its commitments. The protocol participants are *Agent of action* X_j and *Agent of SPA-node* F_k immediately following the action X_j . After accomplishment of the previous activity of own commitments, the *Agent of robot* R_i having the operational status *available* reads its next action to perform from the ordered list of its commitments and sends the proactive message to the *Agent of the next action* X_j it has to perform querying the status of its precondition.

If this precondition status value is *satisfied* then the *Agent of action* X_j sends confirmation message to the robot R_i and the latter begin to perform the assembly action X_j while informing the *Agent of action* X_j by the corresponding message. When the *Agent of action* X_j has the latter message received the value of the attribute of its internal state *Status of action* transits from the value *not started* to *in progress*.

If the action X_j is not ready yet for performance (in this case, the precondition of *Agent of* this *action* is assigned the status value *not satisfied*) *Agent of robot* R_i is waiting for the message from *Agent of action* X_j confirming its readiness and starts to perform it when has got it. When the robot R_i has the action X_j accomplished it sends the message with the corresponding contents to the *Agent of action* X_j and the latter change the value of its attribute *Status of action* from *in progress* to *accomplished*. Afterwards, it sends the message to the own immediate follower

that is the *Agent of SPA-node* F_k to inform it about action accomplishment to let it change its internal state accordingly.

In the sequence diagram depicted in Figure 5, a message from the*Agent of action* X_j to the *Agent of leader* is shown too, see the dotted arrow on the left-hand side. *Agent of action* X_j sends this message in case, if the time left from the start of the action X_j performance exceeds a predefined threshold value. In fact, this message informs the *Agent of leader* about exceptional situation that can be caused by fault of robot R_i, for instance. It is worth to emphasize here that this message demonstrates an example of proactive behavior of autonomous agent specified in terms of FMS with internal state.

Let us remind that two tables usually specify the formal model of FSM. The first one is *Transition table* of the FSM representing transitions of its internal states depending on input messages (events) and *Table on output messages* generated by FSM in response to the same input messages. These tables are fully developed for all classed of agents performing the aforementioned protocol. They are not shown due to deficiency of the paper space.

7. Demonstrational Example

Let us assume a particular assembly case for three robots $\{R_1, R_2, R_3\}$ to perform assembly process of a product comprising 5 types of assembly units $D = \{D_1, D_2, D_3, D_4, D_5\}$, at that the assembly process includes 12 assembly operations $X = \{X_1, X_2, \ldots, X_12\}$. In each assembly operation, a unit of a specific type out of the set $\{D_1, D_2, D_3, D_4, D_5\}$ is used. Each robot can operate with a limited types of assembly units and their capabilities are presented by three subsets of admissible operations $D(R_1) = \{D_1, D_3, D_4\}$, $D(R_2) = \{D_2, D_3, D_5\}$, and $D(R_3) = \{D_1, D_2, D_5\}$ for robots R_1, R_2, and R_3, respectively. The mapping of assembly unit types to the particular type of assembly operation is given in Table 1. Let us also assume that the assembly units are located in three "baskets" and each robot can access only to its own basket.

Table 1
Correspondence between assembly operations X_i and assembly unit types D_j

X_i, i = 1(1)6	X_1	X_2	X_3	X_4	X_5	X_6
Type of assembly units D_j, j = 1(1)5	D_4	D_5	D_5	D_4	D_3	D_1
X_i, i = 7(1)12	X_7	X_8	X_9	X_{10}	X_{11}	X_{12}
Type of assembly units D_j, j = 7(1)12	D_2	D_1	D_1	D_2	D_1	D_3

Let the set of assembly operations X be partially ordered as depicted for the action scenario in Figure 1 and formally this partial order is specified by the equality (1). Accordingly, Figure 3 presents the expanded scenario graph of the action scenario.

Let us remind that, usually, autonomous group control task includes two subtasks. The first of them is allocation of the robots over the actions of the scenario taking into account their particular capabilities (planning task, for short) and scheduling of the assembly scenario performance. The solution of this task is the set of pairs $< X_j, R_i >$ and time intervals of the corresponding assembly action performances.

The second subtask is real-time autonomous group control, i.e. distributed self-organizing coordination of the robot assembly actions according to the group control protocol specified in previous section (see Figure 5). In general case, other subtasks have to be solved too. An example is the subtask if the robots can exchange assembly units in case the latter is absent in their own baskets or send corresponding requests to the *Agent of leader* asking for the needed assembly units.

The approaches and algorithms intended to solve the first task form special topic in the group control scope and they are out of the paper problems. However, in the numerical example described below, a heuristic planning algorithm working well is used. The heuristic itself determines the local policy on (1) how to select the next scenario action $X_j \in X(> t)$ to plan at the forthcoming iteration step and (2) which robot, among admissible ones, according to their capabilities and consistent with their current plans of workload, to allocate to the selected action. The heuristic policy used is (i) to select the most labor-intensive and ready to perform action situated on the critical path of the action scenario graph and (ii) to allocate the admissible robot performing the selected assembly action for the shortest time interval. Let us note that, if the number of assembly actions is too large, the Contract Net protocol standardized by FIPA [14], or another type of auction protocol can be used, because this class of algorithms demonstrate good quality of solution and computationally efficiency. For the numerical example of this section, the solution computed based on the aforementioned heuristic planning algorithm is depicted in Figure 6 in the form of Gant diagram.

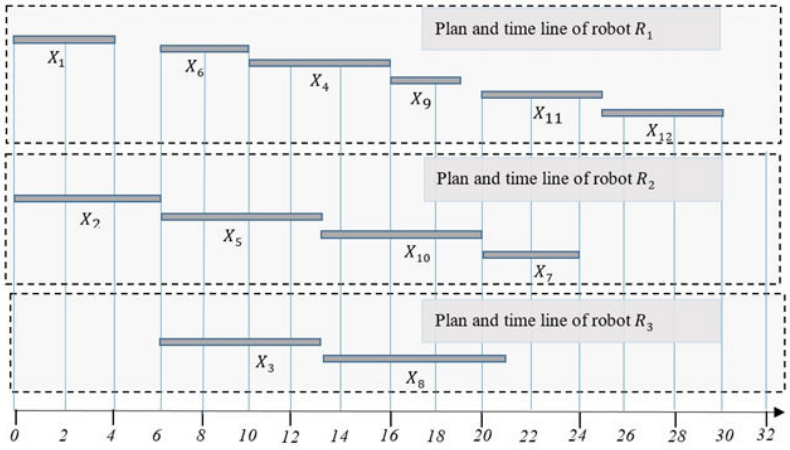

Figure 6: Gant diagram representing allocation of robots over the scenario actions and time line of assembly operation performance

Let us describe the key properties of autonomous real-time control of individual robots' behaviors intended to implement the goal-oriented group behavior scenario represented by Gant diagram (Figure 6) allocating the responsible robot to each assembly action, and timeline of operations, for each particular robot. Let us remind that each robot "knows" own sequence of assigned assembly actions and timeline of their performance, according to their commitments.

Real-time performance group control can manage two types of situations[1]:

(i) normal performance if group behavior follows the pre-designed order with low deviations of times predefined by Gant diagram; in the situations of this type, the control task is to coordinate and synchronize the robots' individual behaviors according to the action scenario and task allocation over the robots;

(ii) non-admissible deviations of real process of performance have occurred.

In the (i)-th case, the agents of the network are operating according to their commitments implementing the real-time group control protocol (Figure 5) as applied to every allocated assembly operation to be performed in the given order. Every such assembly operation is implemented through local interactions of autonomous *Agent of* responsible *robots* and related *Agents of actions* and *Agents of SPA – nodes* (see Figures 4-6). While performing group control protocol, the *Agents of robots* do not interact directly. Their interaction is indirect– through change of external environment, e.g. through change of internal states of *Agents of actions* thus indirectly informing the other related entities about accomplishment of this or that action. Figure 6 shows that at a time instant, as a rule, several robots are operating in parallel. E.g., during the time interval (6, 19) all the three robots are operating concurrently in parallel.

Let us note that Group Control Protocol depicted in Figure 5 implements a variant of self-organizing group control algorithm based on indirect interactions of agents. This kind of self-organization is known as *stigmetry*. It is a form of self-organization exemplified by many bio-inspired self-organization systems. It is also worth mention that, in the developed architecture of self-organizing group control architecture, situational awareness can be also achieved through direct access to the ESG data structure, which, in this case, plays the role of blackboard data structure. The last variant can be preferable if the robots have stable connectivity with it.

In (ii)-th situation, an agent allocated the role of *Leader* is responsible for (1) detection of such situations, (2) revision of the robots' allocations over the rest of the scenario actions $X_j \in X(> t)$, (3) implement Joint intention protocol to renew the agents of robots commitments and conventions and (4) initiate continuation of the group control activity according to the revisited allocation.

8. Brief analysis of Related Works

The paper scope is multi-agent model and distributed algorithm for autonomous group control. The theoretical basic of this research problem is being developed since the middle of 1980^{th}. To the middle of 1990^{th} several theories of teamwork and control models over group behavior and supporting software tools were developed. However, only two of them have constituted the basis for subsequent approaches to and frameworks for the later theories, models and software tools. They are *Theory of Joint Intentions* [15] and *Theory of Shared Plans* [16]. Both theories have sound mathematical foundations; however, no one of them has serious software implementations of deeper level than simple prototyping.

[1]Exceptional and emergent situations as well as their real-time processing constitute an important scope of the group control; however, these tasks are out of the paper objectives.

Two models supported by well known software tools STEM/Teamcore [17] and RETSINA [18] were being developed during that time. However, it was finally found out that both of them possess very weak expressiveness and very low computational efficiency. Other frameworks and models were proposed during that period of time too, e.g. COLLAGEN, GRATE, ADEPT, COOL, etc., however there is no information in the literature about their practical use.

Since 2000 and up to 2010, one can observe a kind of stagnation in the collective behavior and group control research. This stagnation during the indicated time is explicitly indicated in the recent publication of the well-known expert in the group control scope [19]. During that time, no novel results were received because the researchers of different countries unsuccessfully tried to adapt the aforementioned models and software tools in a practically implementable technology.

However, one can observe a definite increase of interest to these researches during the recent decade both in Russia and abroad. This interest was inspired by novel trends in collective robotics as applied to manufacturing, Internet of Things, digital economy of Industry 4.0, for instance. This trend is emphasized in the recent overview [19] published in 2020.

In Russia, such researches are conducted too, although not so actively as abroad. Perhaps, the leading theoretical results in collective robotics are published in [20], [21], [22], [23], [24], [25]. However, practically all of them are developing the ideas of *swarm robotics* that, in fact, has another scope of practical implementation than behavior-based scenario models of group control that is the subject of this paper. Exclusion is presented in [26], [27] proposed the hierarchical scenario knowledge base for group control model. A specific feature of these works is a more general problem statement assuming existence of an adversary counterparty in shared environment and real-time knowledge-based online search for the best behavior pattern to resist an adversary. However, in manufacture-oriented collective robotics no adversary exists.

9. Conclusion

The paper proposes a methodology, multi-agent architecture, state-machine-based formal model for distributed self-organizing group control as applied to autonomous assembly process implemented in a collective robotics scenario. This application problem is of the topmost importance in the modern digital manufacturing.

The main paper contributions are

- problem statement of the assembly process to be implemented by a group of autonomous robots performing assembly scenarios without external interventions;
- conceptual and mathematical model of autonomous distributed group control specified formally as a set of interacting state machines with internal states;
- multi-agent architecture of autonomous collective robotics-based manufacturing assembly system;
- domain-independent distributed algorithm (protocol) implementing real-time self-organizing group control of robot team operating according to a pre-designed assembly scenario.

In fact, the developed Group control protocol is domain-independent and, thus, can be the basis for many other classes of applications. One of such applications is group control of unmanned agricultural machinery.

Future works should be to investigate the particular features of the proposed group control model and algorithm and their influence on control algorithm computational complexity and scalability. Among these features, the priority will be done to (1) cardinality of the set of assembly operations, (2) complexity of partial order relation given over this set, (3) complexity of the assembly operations themselves, (4) cardinality of the set of assembly robot team, and (5) distribution of robot capabilities over the robot team members.

Acknowledgments

The work is being supported by the Ministry of Education and Science of the Russian Federation in the framework of contract agreement No. 075-15-2019-1691 – the unique ID number is RFMEFI60419X0224.

References

[1] J. Ferber, Foreword, in: M. Cossentino, V. Hilaire, A. Molesini, V. Seidita (Eds.), Simulation, Modeling, and Programming for Autonomous Robots, Springer, 2014, pp. V–VI.

[2] J. Odell, Objects and agents compared, Journal of Object Technology 1 (2002) 41–53.

[3] P. Martinet, B. Patin, PROTEUS: A platform to organize transfer inside French robotic community, in: Proceedings of the 3rd National Conference on Control Architectures of Robots (CAR08), Bourges, France, 2008. Https://pagesperso.ls2n.fr/ martinet-p/publis/2008/CAR08bruno.pdf.

[4] S. Lemaignan, R. Ros, L. Mösenlechner, R. Alami, M. Beetz, ORO – a knowledge management platform for cognitive architectures in robotics, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2010), Taipei, Taiwan, 2010, pp. 3548–3553.

[5] S. Dhouib, S. Kchir, S. Stinckwich, T. Ziadi, M. Ziane, RobotML, a Domain-Specific Language to Design, Simulate and Deploy Robotic Applications, in: I. Noda, N. Ando, D.Brugali, J. J. Kuffner (Eds.), Simulation, Modeling, and Programming for Autonomous Robots, volume 7628, Springer, 2012, pp. 149–160.

[6] V. Gorodetsky, V. S. V.V., D. Trotskii, The Reference Ontology of Collective Behavior of Autonomous Agents and Its Extensions, Journal of Computer and Systems Sciences International 54 (2015) 765–782.

[7] M. Tenorth, M. Beetz, Representations for robot knowledge in the KnowRob framework, Artificial Intelligence 247 (2015) 151–169.

[8] X. Li, S. Bilbao, T. Martín-Wanton, J. Bastos, J. Rodriguez, SWARMs ontology: a common information model for the cooperation of underwater robots, Sensors 17 (2017) 569–588.

[9] N. Jennings, Controlling Cooperative Problem Solving in Industrial Multi-Agent Systems Using Joint Intentions, Artificial Intelligence 75 (1995) 195–240.

[10] G. Lambert, Grand Challenges of Information Fusion, in: Proceedings of International Conference on Information Fusion (IF2003), Cairns, Australia, 2003, pp. 213–220.
[11] A. Steinberg, C. Bowman, F. White, Revisions to the JDL Data Fusion Model, in: Proceedings of A1AA Missile Sciences Conference of Naval Postgraduate School, CA, 1998.
[12] Y. Fischer, On Situation Modeling and Recognition, in: Proceedings of the 2009 Joint Workshop of Fraunhofer IOSB and Institute for Anthropomat-ics, Vision and Fusion Laboratory, 2009, pp. 203–215.
[13] P. Salmon, N. Stanton, G. Walker, D. Jenkins, Distributed Situation Awareness: Theory, Measurement and Application to Teamwork, CRS Press, 2017.
[14] FIPA Contract Net Protocol Spec, http://www.fipa.org/specs/fipa00029/SC00029H.html, 2002. Accessed: 20.06.2020.
[15] P. Cohen, H. Levesque, Teamwork, Nous (1991) 487–512.
[16] B. Grosz, S. Kraus, Collaborative Plans for Complex Group Actions, Artificial Intelligence 86 (1996) 269–357.
[17] M. Tambe, Towards Flexible Teamwork, Journal of Artificial Intelligence Research (1997) 83–124.
[18] K. Sycara, M. Paolucci, M. Velsen, J. Giampapa, The RETSINA Multi-agent Infrastructure, Autonomous Agents and Multi-agent Systems 7 (2003) 29–48.
[19] K. Geihs, Engineering Challenges Ahead for Robot Teamwork in Dynamic Environments, Applied Sciences 10 (2020). Https://doi.org/10.3390/app10041368 (Accessed: 20.06.2020).
[20] V. Karpov, Collective behavior of robots. Desired and real. Modern Mechatronics, in: Proceedings of Russian Scientific School, 2011, pp. 35–51. (In Russian).
[21] V. Karpov, Models of Social Behavior in Group Robotics, Control of Complex Systems 59 (2016) 165–232. (In Russian).
[22] I. Kalyaev, A. Gaiduk, S. Kapustiyan, Models and Algorithms of Collective Behaviors in Groups of Robots, Phismatlit, Moscow, 2009. (In Russian).
[23] A. Kulinich, Teamwork model of agents (robots): Cognitive Approach, Control of Complex Systems 51 (2014) 174–196. (In Russian).
[24] V. Pavlovsky, E. Kirikov, V. Pavlovsky, Modelling of Behavior of Big Group of Robots in the Environment with Obstacles, in: Proceedings of the Conference "Control of network – centric and multi-agent systems", St. Petersburg, Russia, 2010, pp. 10–13. (In Russian).
[25] V. Karpov, I. Karpova, A. Kulinich, Social Communities of Robots, in: Series "Science about Artificial", volume 19, URSS, 2019. (In Russian).
[26] B. Fedunov, Intelligent agents in knowledge bases of on-board consulting expert systems in typical situations of operation of human-centric objects, Transactions of Russian Academy of Sciences. Control Theory and Systems 6 (2019) 90–102. (In Russian).
[27] B. Fedunov, On-board Intelligent Systems of Tactical Level for human-centric objects, DeLibry Publishers, Moscow, 2018. (In Russian).