SIMULATION BASED CONTRACT MANAGEMENT IN INTELLIGENT ROBOTIC AGENT SYSTEM

Witold Jacak, Karin Pröll
Department of Software Engineering
Upper Austria University of Applied Sciences
Hagenberg, Softwarepark 11, Austria
Witold.Jacak@fh-hagenberg.at, Karin.Proell@fh-hagenberg.at

ABSTRACT

A multiagent robotic system consists of a group of autonomous agents on the first - lower - level of its hierarchy, and the contract and conflict management agents on the second - upper - level of the hierarchy. The contract management agent considers task distribution when a new job enters the system. It has to direct autonomous agents by specifying individual goals (subtasks) for each of them. The implementation of the simulation based contracting process for transfer jobs is presented.

INTRODUCTION

Generally, a multi-agent system is a community of independently acting agents working together to achieve a common goal which is beyond the individual capabilities or knowledge of each agent. More recently, the term multi-agent system has been given a general meaning, and it is now used for all types of systems composed of multiple autonomous components showing the following characteristics (Cohen and Levesque 1995; Brenner *et al*. 1998; Haddadi 1995; Sandholm 1999): each agent has incomplete capabilities to achieve a common goal, there is no global system control, data is decentralized and computation is asynchronous. An application of such a system is a multi-agent based production system.

Small batch production requires manufacturing and complex control systems with reasonably high flexibility, not only in manufacturing equipment, but also in planning, scheduling, handling, and management decision making. Technological process planning for a mechanical product involves the preparation of a plan that outlines the routes, operations, machine tools, fixtures, and tools required to produce the part.

The distribution of operations on agent acting in the global production system is performed by the job dispatching system. Only jobs ready for processing are considered and transferred to the dispatching system (Jacak 1999; Andersson and Sandholm 1999). Not all waiting jobs can be executed simultaneously. The job dispatcher is constructed according to different principles, depending on the component intelligence in the agent system. Very popular in industry is a dispatching system, which concentrates decision-making power in the higher system level, i.e. the dispatching level. A distributed system based on a job contracting mechanism moves the computational effort usually carried out by the dispatcher of a multi-agent system to the agents themselves. An agent is able to plan the action in real time. As a consequence the dispatcher in a distributed agent system determines a subset of waiting jobs for processing.

In the paper we focus our consideration on the contracting problem of *part transfer actions* between different workstations which are serviced by the robotic agent system. Such actions are called *Pick and Place* operations (Jacak 1999; Proell 2002).

In contrary to the centralized approach for job dispatching, robotic agents are contracted by the *contract manager*, which obtains the pick and place positions for transfer operations from the dispatcher. Each agent should then plan a safe motion to perform the transfer operation and to calculate the costs of this action. This planning and calculation is done by simulation. For this the geometrical model of the agents environment and the kinematics model of the technical component (robot) of the agent are needed. The result of the simulation process is then returned to the contract manager, which decides about assigning a transfer operation to individual agents.

MULTI AGENT ROBOTIC SYSTEM ARCHITECTURE

The group of operational agents (robotic agents) establishes the first level of the multi agent system - the *operational* level. Intelligent operational agents represent entities that independently perform requests on demand of the super level. The management agents populate the super level of the multi agent system, called *management* level.
The main goal of a multi agent system is to solve a common task, which is split up into several subtasks that are distributed by the management level to individual operational agents in the system. Goal distribution on the one side and task performance on the other side require two different units of cooperation and negotiation in the multi

agent system. Therefore the super level contains two agents *Contract Manager* and *Conflict Manager*.
The *Contract Manager* cares about goal distribution when a new job enters the system. It has to manage all operational agents by specifying individual goals (subtasks) to them. This can be done in two different ways. The *Contract Manager* can distribute goals to agents in a hierarchical way by simply assigning a goal to an individual agent, or he can offer a goal as a service request to the whole operational agent community. This entails a bidding process between contract manager and agents in a market-like style (Brenner *et al.* 1998; Sandholm 1999). The hierarchical structure of multi agent system is shown in Fig. 1.

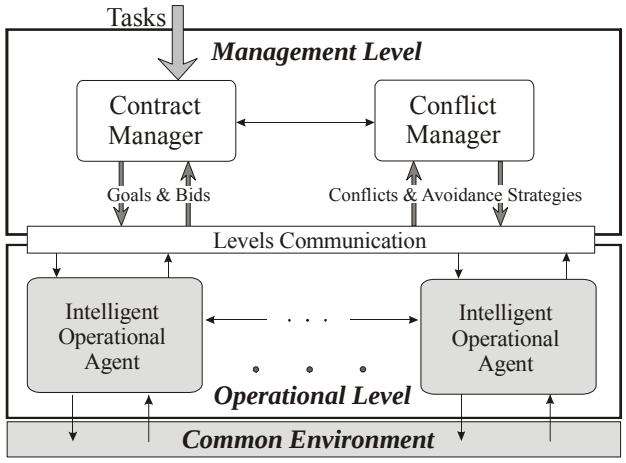

Figure1: Multi agent system

The second management agent - *Conflict Manager* - cares about cooperation and negotiation while each operational agent performs its assigned goal. As the individual behavior of all agents involved in goal achievement cannot be predicted in advance, the goals of two or more agents can be in direct conflict with one another and the achievement of the common goal is endangered. In order to resolve a conflict situation, a negotiation process among all conflict parties has to take place. The result of the negotiation should be a solution, which results in goal achievement for all involved agents (Proell 2002).

Intelligent Robotic Agent

The structure of an intelligent robotic agent is determined by specialized software and hardware components. The basic component of a robotic agent is the robot, equipped with manipulator, effectors and different sensors.
The components are connected via an internal communication bus. The communication between components occurs by exchange of messages, coupled with respective protocols of a communication session. The intelligent robotic agent consists of cooperating components, such as:

- *Technical Component* is a hardware component (robot), which consists of a manipulator, sensor systems, and drive control system,
- *Action Planning and Execution Component* is a component, which first simulates and then plans motions and actions of the technical component and executes them via the drive control system,
- *Action Safety Protection Component* is a component, which tests the safety of movements and prepares the reaction in case of dangerous actions,
- *Communication and Negotiation Component* is a component, which coordinates communication with other agents and the super level, and performs the negotiation process from the agents point of view,
- *Knowledge Base Component* is a component, which contains the different models of knowledge representation.

The structure of a robotic agent system is shown in Fig. 2.

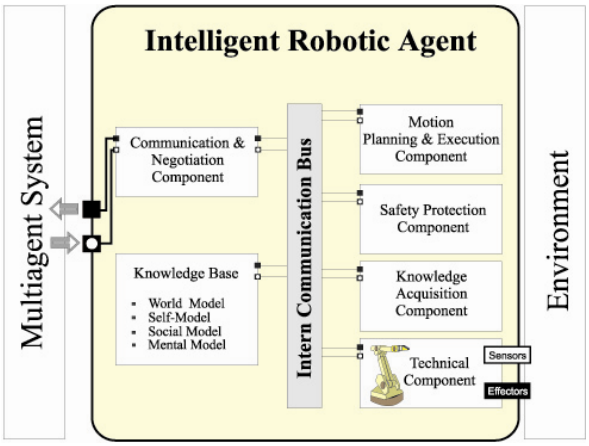

Figure 2: Structure of the robotic agent

The components build the intelligent robotic agent, which is able to perform complex actions in a partially known world. The components are grouped in a layered architecture (Proell 2002; Jacak *et al*. 2001) comprising four layers: the action execution layer (hardware component - robot manipulator), the behavior-based layer (safety protection component) , the local planning layer (action planning component), and the cooperative planning layer (negotiation and communication component). As we have mentioned previously, each agent has its knowledge base, i.e. the set of models and methods needed for decision making. The typical knowledge base contains different models and their formal representations: a *world* model, a *social* model, a *mental* model and a *self* or *plant* model.

Social model – Group Knowledge

Group knowledge of agents cumulates on experiences gathered during execution of different actions in the system. Compilation of group knowledge can be considered as a an agent's life long learning process. It is the precondition for collaboration of agents in order to find subcontractors for common task achievement.
For establishing the set of the possible partners each agent can store in its social model: the identifier of previous conflict partners, the coordinates of the location where the conflict took place, called contact position $q_{contact}$ (state in its state space Q) and the contact distance $d_{contact}$ in the conflict situation. These agents are potential partners for complex transfer actions. One agent can get into conflict with the same other agent many times, but the distance to it can vary. It seems obvious that only the contact position with minimal distance should be stored..
It is possible that the different agents leave the common environment and the social model is incomplete. For this reason is necessary to remember the time of the last contact $\tau_{contact}$, additionally.
The *i*-th agent represents its group knowledge, called *Collaboration,* as the initially empty set of tuples

$$Collaboration = \{(neighbor\_agent, q_{contact}, d_{contact}, \tau_{contact}, q_{transfer})\} \quad (1)$$

where:

- *neighbor_agent* is the identifier of the agent with which conflict occurred,
- $q_{contact}$ is the internal state of the agent at conflict occurrence,
- $d_{contact}$ is the distance to the agent with which conflict occurred,
- $\tau_{contact}$ is the time of the conflict occurrence, and
- $q_{transfer}$ is internal state which makes part transfer possible. his parameter is initially empty. The transfer position is calculated based on simulation and stored when both agents try to prepare a common bid for the contract manager.

The contract manager announces every waiting transfer action to all agents. The announcement is represented by a message which contains the Pickup (start) and Place (finish) pose (position and orientation of the effector) of the Pick & Place operation.
All agents check whether both poses lie in their individual service spaces. In that case it is not necessary to find collaboration partners in the agent group. Single agents can calculate their bid for the transfer action and send it to the contract manager.
If only one position lies in the agent's service space, the execution of the transfer action is not possible for one single agent but requires cooperation between agents. A team building process has to take place which presumes that each agent has to have knowledge about agents, which are physically close to itself.
The agent uses its social model to determine which agents it could collaborate with. The partner searching process starts with sending a message to all agents, annotated in the collaboration set in its social model. It queries them whether the final point of the transfer action lies in their service space or not. This process continues, until either a sequence of agents is found that can collaborate on this task and the last agent achieves the final state, or no such sequence is possible.

Mental model

For secure realization of a preplanned action, we propose the introduction the security zone ε, which determines a safety distance between the agent's current state and the state of an adjacent agent. When the security zone is penetrated a conflict situation occurs and the agent applies methods of the mental model in order to determine its behavior for circumvention of the penetrating object.

World model

The knowledge represented here is the geometrical model of the robotic agent's environment (Jacak 1999). Many different methods can be used for geometrical representation of the agent service space. The model geometry of each object can be created by a solid modeling incorporating the design and analysis of virtual objects created from primitives of solids stored in a geometrical database. Constructive solid geometry handles primitives of solids, which are bounded intersections of closed half-spaces, defined by planes or shapes. More complex objects can be built by using set operations, such as union, intersection and difference of solid bodies. Such methods can be used for modeling the artificial world but are not preferred for modeling the dynamical scene based on sensor information.
To obtain fast and fully computerized methods for collision detection we use additional geometric representation for each static object based on ellipsoidal representation of 3D objects using ellipsoids for filling the volume. The ellipsoidal representation of a static object is convenient to test collision freeness of agent configurations. It can be reduced to the "broken-line ellipsoid" intersection detection problem, which in this case has an easy analytical solution.
To obtain a uniform representation of the robotic agent and its environment, a transformation of the geometrical model of the work-scene from the base Cartesian frame into the joint space of the agent often is performed (Jacak 1999).

Plant-model: Model of the agent's hardware component

The plant model contains the knowledge about construction, properties and structure of the hardware component. Here, knowledge of the kinematical properties of the robotic agent is provided in order to decide about collision avoidance mode and avoidance path. Therefore, the forward and inverse kinematics models of the robotic agent should be known. These models can be created in different ways using different formal tools as e.g. symbolically computed models, numerical models or a neural network-based model (Jacak 1999; Proell 2002). For our simulation based action planner we use a finite state machine model for the agent actor.

In order to make it possible to calculate and simulate safety behavior of the agent with respect to obstacles or to other agents locations, it is necessary to create the skeleton model of robot depending on the state q. The state $q \in Q$ should determine the position of the manipulator, called the manipulator's configuration in real space. The most frequently used state description of a robot with n-DOF is its representation as a vector of joint variables: $q=(q_i|i=1,..,n)^T$ where $q_i \in [q_i^{min}, q_i^{max}]$ is a range of changes of the i-th joint angle. Recall that the i-th joint's position in Cartesian base space, assuming that all the joint variables q_i are known, is described by the Denavit-Hartenberg matrix (Jacak 1999).
The most suitable model for the hardware component of is a discrete dynamic system defined as

$$TC=(Q,U,Y,f,t) \quad (2)$$

where Q denotes a set of inner states of the agent actor and the state $q=(q_i|i=1,..,n)^T$, U denotes a set of input signals, An output Y represents a skeleton model of the agent's manipulator described as the vector $y=(P_i|i=0,1,..,n)^T$ where P_i is the point in the base coordinate frame describing the current position of the i-th joint. The function $f:Q\times U \rightarrow Q$ is the one step transition function of the form $q(k+1)=f(q(k),u(k))$ and $t:Q\rightarrow Y$ is an output function.

One possibility for constructing such a model of robot's kinematics is based on an arbitrary discretization of angle increments of the agent mechanical joints (Jacak 1999; Proell 2002).
Using this fact all angles can change only by a definite increment. We define the input set U for the model as: $U= \times\{u_i|i=1,..,DOF\}$, where $u_i =\{-\delta q_i, 0, \delta q_i\}$ is the set of possible (admissible) directions of changes of the i-th joint angle.
Having defined the set U, it is possible to describe the changes of successive configurations of the agent's link as discrete linear system of the form:

$$q(k+1)=q(k) + \Lambda\, u(k) \quad (3)$$

where $u(k)$ is the vector of increments of angles of the joint and Λ is diagonal matrix describing the length of the angle's step changes at each joint.
In order to make it possible to check the configuration with respect to obstacles locations in world model, it is necessary to create an output function. As we have stated previously, the agent manipulator's position in the base frame is represented by a skeleton of an agent arm (Jacak 1999). Recall that the i-th joint's position in Cartesian base space, assuming that all the joint variables q_i are known, is described by the Denavit-Hartenberg matrix [3] $T_i= A_1\, A_2 \ldots A_i$ where A_i is the transformation matrix between the coordination frame E_i of the i-th joint and the coordination frame E_{i-1} of the $(i-1)$-th joint. The last column of the matrix T_i can be used to determine the output function of model as:

$$t(q(k))=(P_o, t_i(q(k))|i=1,..,n)^T \quad (4)$$

where $t_i\,(q(k))= P_i$ is the element of the last column of the matrix T_i.

MANAGEMENT LEVEL: CONTRACTING

In modern industry we observe the tendency to distribute decision making to all system components and the robotic agents get more autonomous and are able to adopt their behavior to different changes in the surrounding environment. This system is called distributed robotic system and it needs only partial knowledge about the overall system behavior. In an extreme case the system has very poor knowledge about the surrounding environment (for example, a system which acts in the real world). In this case the system components (i.e. robotic agent) must be extremely autonomous and should be able to plan and execute actions in a reactive manner.

As a consequence the dispatcher in a distributed robotic system can choose a subset of waiting jobs which is possible to be realized. In contrary to the centralized structure of job dispatching, autonomous agents are contracted by the contract manager which obtains only information about the pick and place positions for transfer operations from the dispatcher.

Each agent itself should then plan a safe motion to realize the transfer operation and to calculate the costs of its action. This information is returned to the contract manager which decides about assigning a transfer operation to individual agents.

Contract management

Decomposition of the waiting jobs into individual transfer actions results in the creation of a queue of waiting transfer

motions. These transfers are the current common task set of the multi agent system.
The contract manager sends the common task set per message exchange to all agents. Each message contains the start position p_s and the finish position p_f of the *Pick and Place* operation. The contracting process is shown in Fig. 3.

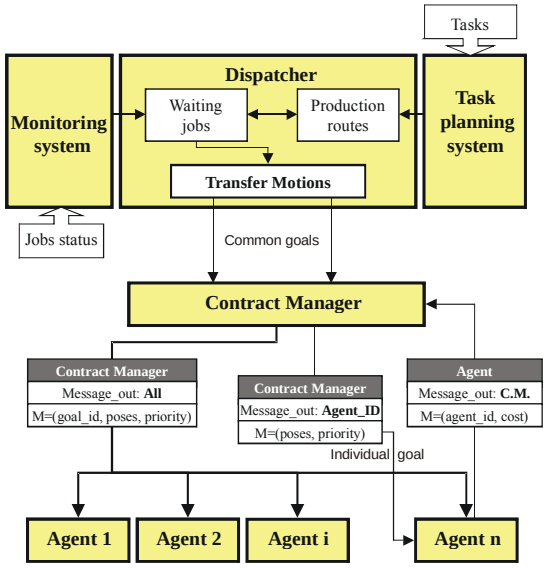

Figure 3: Contracting process

The contract manager sends messages to all agents with

message_out = *(Receiver:* ***All****, task_id, poses =(p_s, p_f), priority)*
Each robotic agent has own queue of waiting tasks. We assume that agent can have only one waiting task in the queue, i.e. the task, which is contracting during the realization of the current transfer motion.
The contract manager awaits the answers of all individual agents together with a cost estimation for goal realization.

message_in = *(Receiver:* ***Contract Manager****, task_id, Agent_id, cost)*

SIMPLE CONTRACTING PROCESS

A positive answer is possible if both points i.e. start and final pose lie in an obstacle free part of the service space of the agent and the considered agent has an empty queue of waiting goals. If the both poses *(p_s, p_f)* lie in the service space, then the agent calculates the time estimation of the motion to realize the goal. For this purpose the motions necessary are simulated and the estimated time for task realization consists three components:

- time which is needed to finish current task τ_{rest}
- time of motion to achieve the start position of the new task $\tau_{prepare}$,
- time of motion to achieve the final position of the new task $\tau_{realize}$.

The cost estimation of task realization of agent

$$cost(agent\_id, task\_id) = \tau_{rest} + \tau_{prepare} + \tau_{realize} \qquad (5)$$

Each of these times can be calculated by estimation of the appropriate length of motions in joint space Δq_i and average angle velocity $avgq'_i$. For example

$$\tau_{rest} = max\{||q_i^{current} - q_i^{final\sim old}|| / avgq'_i \mid i=1,..,DOF\} \quad (6)$$

This estimation is only a rough approximation of the motion time. In case of concurrent realization of motions of many agents it is necessary to simulate and plan a conflict- and collision free motion for all involved agents. It can be done by motion planning methods using a full known environment model for the simulation (Proell 2002; Jacak *et al*. 2001). Simulation and planning is done by the conflict management agent, which uses kinematics and the dynamics models of the robotic agents and a geometrical model of the work space (Jacak 1999; Proell 2002). In case of conflicts the planner changes the motions of the robotic agents. The new conflict free paths of motions are much longer as the independently simulated motion trajectories for every agent. Based on the dynamics model of the robotic agents it is possible to calculate the optimal time of motion for all agents (Jacak 1999). For a simple cost approximation it is enough to calculate the motion time with average angle velocity *avgq'* for each join.

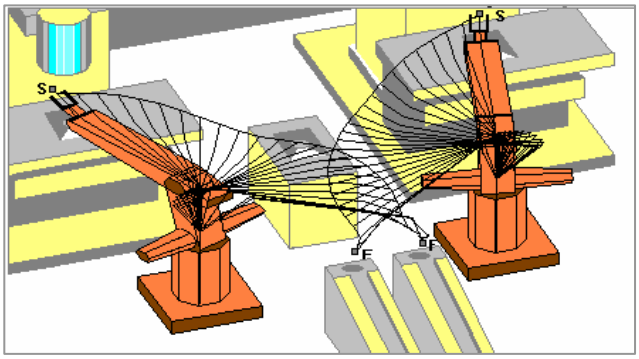

Figure 4a: Conflict trajectories of motions.

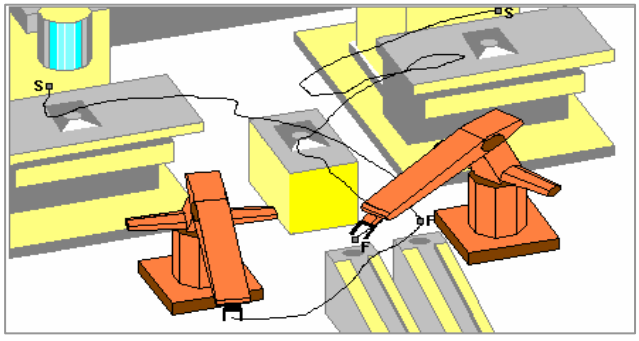

Figure 4b: Conflict free trajectories of motions

For example in case of concurrent realization of independently calculated motions of robot in Fig. 4a a conflict situation occurs and new conflict free paths must be calculated. (Fig. 4b), with longer realization times (Fig. 4c).

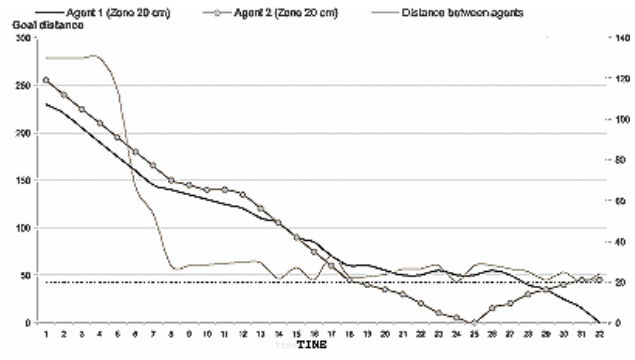

Figure 4c: Time of motions along conflict free trajectories

SIMULATION BASED SUBCONTRACTING PROCESS

The contracting process can be extended in case of just one point of the transfer action (start position) being in the service space of an agent. Then this agent can start the subcontracting process.
The searching process for team members is initiated by sending a message to all agents stored as group knowledge *Collaboration* in the *social model* of the agent under consideration. The agent uses its social model to determine which agents it could collaborate with: these are all agents the given agent has previously been in conflict with.
This message contains information about the final point of the given transfer motion.
When a positive answer is sent by one or more agents, then the agent initializing the subcontracting process sorts potential partners according to minimal distance between known conflict positions and the given start position, in order to select only those agents for partners which are closer to the given start position. Next the search for finding a proper transfer position and orientation for the transfer operation must start.

Transfer position $q_{transfer}$ search.

The simulation based action planner of the agent generates the new movements to find a proper transfer position using the knowledge contained in the *Collaboration* set by the following steps:

Step 1: If $q_{transfer}$ in the element $\{(neighbor\_agent, q_{contact}, d_{contact}, \tau_{contact}, q_{transfer})\}$ of *Collaboration* set is not empty then only the modification of fine motion parameters are performed, i.e. $q_{transfer}$ obtains additionally displacement for the transfer action, depending on the shape of the transferred part.

If $q_{transfer}$ is empty then the action planner requests the current positions of partner agents and tries to plan the motions for both agents from the $q_{contact}$ of first agent and current position of second agent such that the distance d between the effectors of both agents are within a small ρ value. The search algorithm uses the geometrical world model and the FSM model of the agent's kinematics and simulates the motions of both agents as in step 2 – step 5.

Step 2: Based on the positions of second agent action planner establishes the parameters for motion simulation based on A* graph search algorithm, such as type of evaluation function and type of agent configuration feasibility testing function and the security zone of motion.
The evaluation function for i and k agents is always in form $e^k(q) = e^i(q) = d_{effector}(q)$ i.e. is the distance function between the effectors ends of both agents. The configuration q is feasible if is collision free and is not in direct conflict with other agents. The direct conflict between agents i and k occurs if $d(q^i, q^k) > \varepsilon$, where d denotes the distance between the geometrical skeletons of both agents. The ε is the minimal distance between agents where direct collision not jet occurs. In this case the ε is respective small, what determines the small security zone.

Step 3: The action planner starts a state-graph searching algorithm A* from the current position of the first agent q^i_c (initially equal to $q_{contact} = q^i_c$), with previously established evaluation and feasibility testing functions.
The searching stops if the distance $d_{effector}(q)$ is equal to ρ or if the OPEN set is empty or if the OPEN set has more as N elements. To simulate the motion steps the planner uses the FSM model of the agent's hardware component.

Step 4: The temporary path of the motion is calculated i.e. $path^i_t = q^i_c \rightarrow q^i_b$ where q^i_b is the best configuration in the CLOSE set. Depending on the length and the conflict freeness of this path the new configuration of the motion is chosen and the motion is simulated.

Step 5: The new current state (configuration) q^i_b is analyzed and an adequate message is send to the partner agent. The message includes the new configuration q^i_b . By sending the message the simulator of the first agent ends his activity and waits for a new message from the partner agent which starts the same simulation algorithm, based on its own current position and the obtained q^i_b of the first agent.
The simulation process (steps 3-5) will be repeated and stopped when the $d_{effector}(q)$ is less than ρ. This last position q^i_b is stored in *Collaboration* set as $q_{transfer}$ position.

Common cost calculation

Next the common cost can be calculated and the collectively bid can be sent to the contract manager. Both agents calculate the estimated time for achieving the transfer position.
The cost for the first agent's movement can be calculated as:

$$\tau^{1}_{tran} = \tau_{rest} + \tau_{prepare} + \tau_{realize} \quad (7)$$

where
τ_{rest} is time which is needed to finish current realized task,
$\tau_{prepare}$ is time of the motion needed to achieve the start position of the new task,
$\tau_{realize}$ is the time of the motion needed to achieve the transfer position.
In the case when the waiting tasks queue of the second agent is empty, the cost of the second agent can be calculated as:

$$\tau^{2}_{tran} = \tau_{rest} + \tau_{realize} \quad (8)$$

where
τ_{rest} is the time needed to finish the current task,
$\tau_{realize}$ is the time needed to achieve the transfer position.

When the waiting tasks queue of the second agent is not empty the τ^{2}_{tran} is extended by the realization time of the task in the queue.
The common costs can be estimated as

$$cost(agent\_id, task\_id) = = max\{\tau^{1}_{tran}, \tau^{2}_{tran}\} + \tau^{2}_{realize} \quad (9)$$

where
$\tau^{2}_{realize}$ is the time needed to achieve the final position by the second agent.
This process continues, until either a sequence of agents is found that can collaborate on this task and the last agent achieves the final state, or no such sequence is possible.
This contracting process is shown in the Fig. 5.

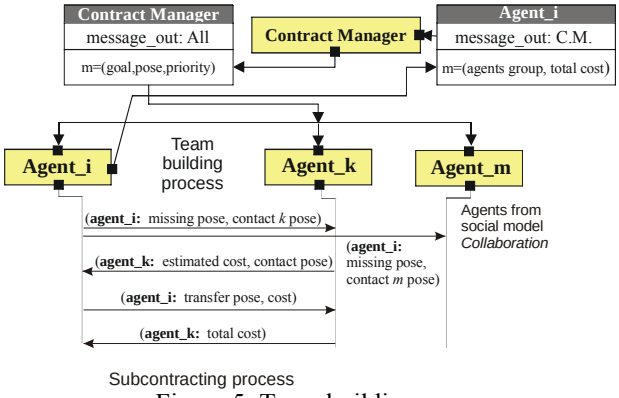

Figure 5: Team building process

An example of the team building and contracting is presented in the Fig. 6. The figure presents the search for transfer position started from, $q_{contact}$ positions of both partners.

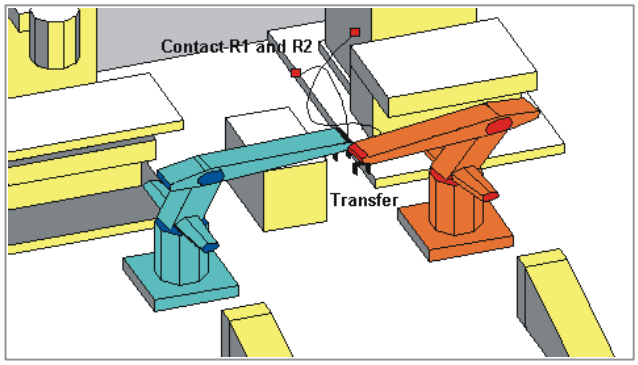

Figure 6: An example of team building: search for a transfer position

CONTRACTING

The Contract manager chooses the best offer based on the estimated cost of task realization and its own scheduling algorithm. To perform the selection of the offers the contract manager should find the one-to-one partial function

$$\sigma : Task_{current} \rightarrow Agents \quad (10)$$

which assigns the *Pick and Place* operations to the agents. More exactly it constructs the partial function with following constraints:

1. There must exist an agent, which has an empty queue of waiting tasks and services the devices realizing the chosen job which is connected with the task (one step scheduling)
2. One agent can realize only one task at a time
3. A maximum number of tasks should be executed simultaneously
4. The selected tasks should have the highest priority
5. The selected tasks should have the minimal summary cost of realization

Some intelligent scheduling algorithms can be found in (Jacak 1999).
Based on the scheduling result of the offers the subset of common tasks are assign to individual agents. This subset $\sigma^{-1}(Agents\_id) \subset Task_{current}$ creates the individual set of agents tasks.
For the agent, where $\sigma^{-1}(Agents\_id) \neq \emptyset$ the task is transferred into the agent's queue of waiting tasks i.e.

$$Task_{current}(Agent\_id) = Task_{current}(Agent\_id) \cup \sigma^{-1}(Agents\_id) \quad (11)$$

The agent can only obtain any new contracts as long as its task queue is empty. The transition graph of the engine of the contract management agent is presented in Fig. 7.

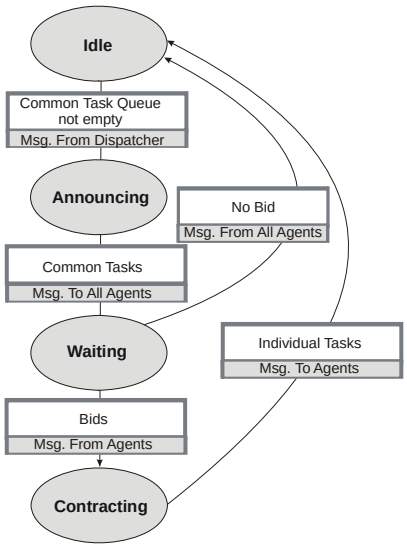

Figure 7: State transition graph of contract manager

An example of team building and contracting is presented in the Fig. 8. It shows the motions for task realization by two cooperating agents.

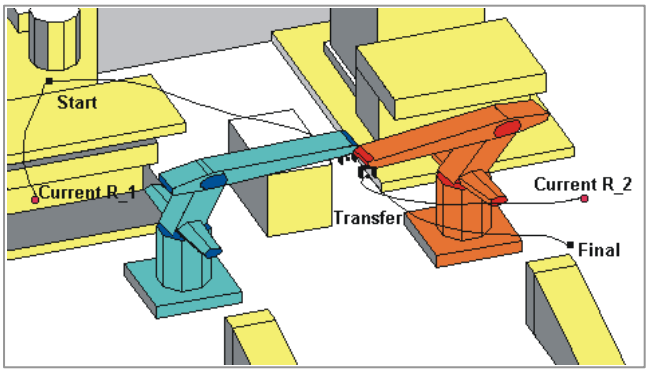

Figure 8: Two cooperating agents realizing task

REFERENCES

Andersson, M. and Sandholm, T. 1999. Sequencing of Contract Types for Anytime Task Reallocation. In Agent-Mediated Electronic Trading, Carles Sierra (ed.). Springer Verlag Lecture Notes in Artificial Intelligence 1571, pp. 54-69.

Brenner W., Zarnekow R., Wittig H., Intelligent Software Agents, Springer-Verlag, 1998

Cohen P. and Levesque H., Communicative Actions for Artificial Agents, Proceedings of the First International Conference on Multiagent Systems, San Francisco, Cambridge, AAAI Press, pages 65-72, (1995).

Haddadi A., Communication and Cooperation in Agent Systems, Springer Verlag, 1995

Jacak W., Proell K., Dreiseitl S., - Conflict Management in Intelligent Robotic System based on FSM Approach Lecture Notes in CS Nr. 2178, 2001 pp. 52-66

Jacak W., - Intelligent Robotic Systems: Design, Planning and Control - Kluwer Academic/Plenum Publishers Publ., New York, Boston, USA, 1999 pp. 316

Proell K., Intelligent Multi-Agent Robotic Systems: Contract and Conflict Management, PhD Thesis, Johannes Kepler University Linz /Austria, 2002

Sandholm, T. 1999. Automated Negotiation. Communications of the ACM 42(3), 84-85.

BIOGRAPHIES

WITOLD JACAK received the M.S.E. degree in electronics and next the Ph.D. degree in control and system engineering from the Technical University of Wroclaw, Poland and the Habilitation degree in intelligent multi agent systems from Technical University of Warsaw, Poland in 1973, 1978 and 1992 respectively. From 1977 to 1989 he was a Professor Assistant and from 1989 to 1995 Professor at the Institute of Technical Cybernetics, Technical University of Wrozlaw, Poland. From 1991 to 1998 he was the Guest Professor at University Linz, Austria. Since 1994, he has been the Head of Information Technology Faculty at Upper Austria University of Applied Science, Hagenberg, Austria. His research interests include artificial and computational and distributed intelligence, multiagent autonomous systems, cognitive modeling and simulation and knowledge engineering. Dr. Jacak is a member of editorial boards and council member of several national and international committees. His e-mail address is: jacak@fh-hagenberg.at

KARIN PROELL received the M.S.C and Ph.D. degree in computer science from the Johannes Kepler University in Linz, Austria in 1982 and 2002 respectively. In 1999, after a several years activity as software engineer in manufacturing industry she has become as a lecturer in computer science at Information Technology Faculty at Upper Austria University of Applied Science, Hagenberg, Austria. Her research interests include computational and distributed intelligence, multiagent autonomous systems and knowledge engineering. Her e-mail address is: proell@fh-hagenberg.at