17

ACQUAINTANCE MODEL IN RE-PLANNING AND RE-CONFIGURATION

Vladimír Mařík, Michal Pěchouček, Olga Štěpánková
Grestner Laboratory for Intelligent Decision Making and Control
Czech Technical University in Prague, Prague, Czech Republic

Abstract: The problems of re-configuration and re-planning in multi-agent systems exploring acquaintance models of their behavior are discussed throughout this paper. Both the autonomous and supervised methods of re-planning are distinguished. The advantages of the tri-base acquaintance models used in the ProPlanT production planning system for solving the re-configuration and re-planning tasks are sketched.

1. INTRODUCTION

The most valuable advantage of planning and scheduling systems based on the multi-agent approach is their robustness and flexibility in dynamic re-planning. The process of re-planning is either invoked by re-configuration of the multi-agent community or it invokes re-configuration by itself (to meet requirements of the user and to achieve a successful plan).

The problems of re-configuration and re-planning in connection with the ProPlanT multi-agent production planning system are discussed in this paper.

2. RE-CONFIGURATION AND RE-PLANNING

2.1 Classification

In our understanding the process of *re-planning* changes an already elaborated plan, which can be carried by a single entity or by more parties. We are concerned with the latter case in the context of multi-agent systems. Re-planning appears here as a natural consequence of system's *re-configuration* which changes the structure of the multi-agent system in terms of change of various relationships among agents. We classify re-configuration and re-planning processes with respect to the time when they happen, to their cause, the effect on the community they has, and the strategy.

Re-planning may occur either in (i) the planning phase of the community lifecycle, (ii) after plans are elaborated and fixed, or (iii) when plans are getting executed. Re-planning in the planning phase of the community is triggered whenever the planning cannot proceed (a part of the plan that depends on another part that failed to be planned). This is a classical problem of backtracking and there is no need to do any research in this area. However the remaining two cases are more interesting and will be commented on throughout the paper.

A requirement for re-planning may be triggered by different types of causes. In principle we distinguish between:

- **request-driven-re-planning** – re-planning caused by a request to incorporate a new high priority plan that could not be achieved unless some of previously elaborated plans are postponed or at least reshuffled somehow, or
- **community-driven-re-planning** – re-planning initiated by certain changes in the multi-agent community, such as:
  - agent leaving the community (failing to provide all services),
  - agent losing a capability (refusing or failing to provide some of services),
  - agent acquiring new capability (providing a new kind of services), or
  - new agent joining the community (new resources become available).

The problem of re-planning addresses real manufacturing problems as it concerns the already fixed plans which should be re-specified into (sub)optimal plans either with respect to a new high-priority project or due to some failure of a manufacturing device. It is very often the case that in the moment of re-planning some parts of the fixed plans have been already manufactured and manufacturing of the given partially processed product should be completed.

If a new high-priority project arrives, which is the case of the **request-driven-re-planning** – we have to find suitable resources for production and we have to consider how to release the capacity of these resources at appropriate times. We have to decide which parts of the other plans can be shifted and how. However, when some part of the production facility is broken, an instance of the **community-driven-re-planning** is invoked. We distinguish two possible situations. First, there are no available substitutions for the broken machines or systems. In such a case, we have no real chance for intelligent rescheduling. In such a case, re-planning can be achieved by shifting the times in the plan only. This situation is more interesting since we can find some possible substitution for the broken system. The operations from the broken equipment have to be re-planned to be completed on another machine. As such a machine is usually occupied by other operations, the decision about releasing the capacity has to be done – similarly to the **request-driven** case. For that purpose, we can balance the priority of the plan, which is re-planed. We can see that priority mechanism can be applied to solve **community-driven-re-planning**. Thus, both cases can be treated in a similar way.

In the real world we can also expect, that in some future time the broken part will be repaired and/or a new machine is involved. This is also an instance of the **community-driven-re-planning**. This type of re-planning can solve many problems with delayed plans and makes the re-planning procedures easier. If we have delays in our plan and the new resource is introduced then some delayed operations can be re-scheduled according to their priority and the delays should be removed.

Each time the multi-agent system is requested to replan the afflicted sub-community (*zone*) of agents involved in re-planning has to be identified. In reference to the **community-driven-re-planning** the afflicted zone depends on the role of the broken agent in the MAS system. Two basic cases have to be distinguished:

- an agent who accounts for decomposition and coordinates a subproject dies or
- an agent leaving the community is one of the lowest level node in the multi-agent hierarchy.

In the former case, the afflicted zone consists of all the subordinate agents (*subordinate zone*) complemented by all agents involved in the planning process waiting for the particular agent (*collaborative zone*). In the latter case, the

collaborative zone shall be identified only. These zones will be defined more precisely in the section on the Tri-base Acquaintance Model.

Moreover, we have to determine how the process of re-planning and re-configuration is organised and managed by agents themselves (*autonomous re-planning*) or by a special agent responsible either for cause detection, or for a re-planning and re-configuration process or for both (*supervised re-planning*).

2.2 Cause detection

In order to trigger the process of re-planning we need some means for detection of inconsistent behaviour of the system. There are several approaches how to detect the failure of a plan.

The first possibility is the standard *execution-monitoring* model originally used in the STRIPS planner. The plan segments are annotated with preconditions. The preconditions of a plan segment are all those preconditions of the steps in the segment that are not determined. The failure is than detected by comparing the preconditions with the current state description.

The second way is the *action-monitoring* model. In this approach the preconditions are checked for each action as it is executed. Such method is simpler and avoids the necessity for annotations thus it fits better the real-world systems with individual action failures.

Both these forms of monitoring require enough information to tell whether a plan is about to fail. Depending on the possibility to detect an unexpected status, we can divide the causes of plan failures into two categories:

- **Bounded indeterminacy**: Unexpected effects can be enumerated and described.
- **Unbounded indeterminacy**: The set of unexpected effects is too large – the case in complex, dynamic domains.

Since we have good knowledge about possible plan failure causes we can assume the bounded indeterminacy. Thus, the re-planning agent is designed to use the description of the situation for execution monitoring and re-planning in the community.

In the following we will illustrate the problem of re-planning in the domain of production planning in the ProPlanT (Production Planning Technology) multi-agent system. The concepts of both the tri-base acquaintance model and the meta-agent have been used for solving problems specified above.

3. PROPLANT MULTI-AGENT SYSTEM ARCHITECTURE

A prototype of the ProPlanT multi-agent system has been implemented for planning and modeling the TV transmitter production process in Tesla TV. Resulting from a thorough production process analysis we have identified specific information units the general production process is based on. In principle we cluster agents into two fundamental super-classes: *intra-enterprise agents* (IAE) and *inter-enterprise agents* (IEE). We distinguish among the following basic classes of IAE agents (see Figure 1):

- **Production Planning Agent** (PPA) is in charge of project planning. It is supposed to construct an exhaustive, partially ordered set of tasks that need to

be carried out in order to accomplish the given project. It contracts PMA agents.

- **Production Management Agent** (PMA) is responsible for the project management in terms of contracting the best possible PA agents (in terms of operational costs, offered delivery time, and current capacity). PMA delegates its responsibility either to another PMA or it conducts work of a group of PA agents contracted for the considered task. In this manner, a multi-level managing structure can be modeled.

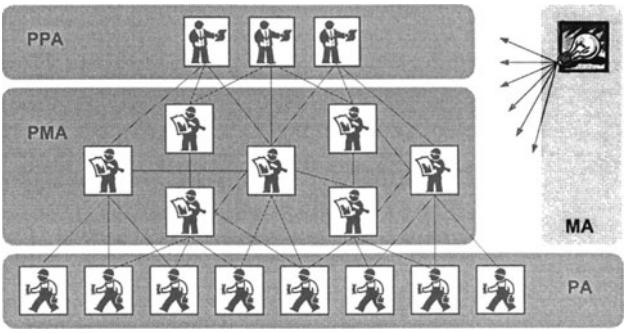

Figure 1 – ProPlanT System Architecture

- **Production Agent** (PA) belongs to the lowest level production units that simulate or encapsulate shop floor production processes on the IAE level. PA carries out the parallel-machinery scheduling of given tasks and manages resources allocation via a special type of database agents. On the IEE level, the PA agent may encapsulate contracted suppliers offering either services or components participating in the manufacturing process. Appropriate optimisation within the community will result in the cheapest (or shortest) production plan.
- Another IEE agent may be a **Customer Agent** (CA). In the current implementation CA agent is the only actor that may trigger the course of production planning. It negotiates with the PPA agent in order to specify the production requirements and both the deadline and budgetary constraints.
- **Meta Agent** (MA) is a special monitoring agent who visualizes information, material and work flows across the agents' community and advises on optimal system's efficiency. It shall be noted that the community of agents will survive well with no meta-agent. 'Ordinary' agents are able to communicate in peer-to-peer manner, but the meta-agent is able to induce specific efficiency considerations from observation of the community workflow.

The CA agents will communicate with the PPA agents only. PPA constructs a component list with the team of PMA agents and delegates further responsibilities to PPAs who contract the best possible PA agents. Final costs and deadlines, a meta-representation of the distributed production plan, will be then back-propagated to the customer.

4. TRI-BASE ACQUAINTANCE MODEL

A agent is recommended to be equipped with a specific knowledge about behavior of the collaborating agents encoded in the *tri-base acquaintance model* [Pěchouček 00] in order to provide an optimal task decomposition. Prior to formalizing the model let us introduce several primitives we will use throughout the course of explanation. Let Θ be a set of all agents within the community and S a set of all tasks the community members are able to decompose. For each $A \in \Theta$ let

- $\alpha(A) \subseteq \Theta$ be an agent's *total neighborhood*, a set of agents an agent A is aware of,
- $\beta(A) \subseteq S$ be the set of all tasks the agent A is able to decompose,
- $\gamma(T)$, contains all possible plans for decomposing the task $T \in S$; a plan for the task T is in the form $\langle T, S, O, C \rangle$, where S is a set of subtasks which ensure completion of the task T provided that their processing meets precedence constraints O and applicability constraints C,
- $\omega(A,T) \subseteq \gamma(T)$ contains those plans for the task T an agent A knows about (if $T \notin \beta(A)$ then $\omega(A,T) = \varnothing$).

The following sets provide time dependent information. Let

- $\varepsilon^t(A) \subseteq \alpha(A)$ be the agent's current *cooperation neighborhood*, a set of agent's A collaborators at the time instant t,
- $\tau^t(A) \subseteq \beta(A)$ contain the tasks being solved by the agent A in a time instance t and the set,
- $\pi^t(A) \subseteq \beta(A)$ be a collection of tasks an agent A is supposed to have pre-prepared in advance in time instance t.

Within the tri-base model each agent maintains three knowledge bases where all the relevant information about the rest of the community is stored. We distinguish among:

- **Cooperator Base** (CB) – it maintains permanent information on co-operating agents (i.e.: their address, communication language, and their predefined responsibility). This type of knowledge is expected not to be changed very often. $\mathrm{CB}(A)$ is then defined as

$$\mathrm{CB}(A) \equiv \{\langle B, \mathrm{Addr}(B), \mathrm{Lang}(B), \beta(B)\rangle\}_{B \in \alpha(A)} \text{ where}$$

  $\mathrm{Addr}(B)$ specifies the agent's B address, $\mathrm{Lang}(B)$ the language it communicates, as already mentioned $\beta(B)$ is a set of tasks the agent accounts for and the set $\alpha(A)$ denotes members of the agent's A scope of the community.
- **Task Base** (TB) – it stores in its *problem section* (PRS) general problem solving knowledge – (i) information on possible decompositions of the tasks to be solved by the agent and (ii) in its *plan section* (PLS) it maintains the actual and most up-to-date plans on how to carry out those tasks, which are frequently delegated to the agent - the owner of the task base, those denoted as $\pi^t(A)$. The formal definition of the $\mathrm{TB}(A)$ is then

$$\mathrm{TB}(A) \equiv \langle \mathrm{PRS}(A), \mathrm{PLS}(A)\rangle, \text{ where}$$
$$\mathrm{PRS}(A) \equiv \{\omega(T,A)\}_{T \in \beta(A)}, \text{ and}$$

$$\mathsf{PLS}^t(A) \equiv \{\langle T, \langle\{\langle s, B\rangle\}_{s \in S}, O, C, \mathsf{Trust}(T)\rangle\rangle\}_{T \in \pi^t(A)},$$

where for any $\langle T, \langle\{\langle s, B\rangle\}_{s \in S}, O, C, \mathsf{Trust}(T)\rangle\rangle \in \mathsf{PLS}^t(A)$ where exist O_1, C_1 such that following constraints are met $\langle T, S, O_1, C_1\rangle \in \mathsf{PRS}(A)$, $B \in \varepsilon^t(A)$, $s \in \beta(B)$ and C is a specialization of C_1 reflecting the considered allocation of the tasks $s \in S$, O is refinement of O_1 and both O and C are valid.

- **State Base** (SB) – stores in its *agent section* (AS) all information on the current load of co-operating agents. This part of the state base is updated frequently and informs the agent who is busy and who is available for collaboration. In the *task section* (TS) there is stored the information on status of the tasks the agent is currently solving. The formal description of the $\mathsf{SB}(A)$ of the agent A is thus

$$\mathsf{SB}(A) \equiv \langle \mathsf{AS}(A), \mathsf{TS}(A)\rangle, \text{ where}$$

$$\mathsf{AS}(A) \equiv \{\langle B, \mathsf{Cap}(B), \mathsf{Load}(B), \mathsf{Trust}(B)\rangle\}_{B \in \varepsilon^t(A)} \text{ and}$$

provided that agent's B capability has the form of $\mathsf{Cap}(B) \equiv \{\langle T, \mathsf{Cost}(T)\rangle\}_{T \in \beta(B)}$, overall agent load is $\mathsf{Load}(B)$, and trust in this information in $\mathsf{Trust}(B)$.

$\mathsf{TS}(A)$ contains relevant information on all the tasks agent A agreed to supervise recently. This set is denoted by $\tau^t(A)$. Formally

$$\mathsf{TS}(A) \equiv \{\langle T, \mathsf{Dec}(T), \mathsf{State}(T), \mathsf{Trust}(T)\rangle\}_{T \in \tau^t(A)},$$

where decomposition $\mathsf{Dec}(T)$ is taken from the $\mathsf{PLS}^{t_1}(A)$ at the moment of the contract (time t1). The $\mathsf{State}(T)$ partitions subtasks from $\mathsf{Dec}(T)$ into three parts: subtasks finished, actually processed, and the rest. The record is complemented with the trust value $\mathsf{Trust}(T)$ denoting the trust in the plan of the task T.

The agent is supposed to select an optimal plan from the $\mathsf{PLS}^t(A)$, where an appropriate amount of plans prepared in advance is stored. By this it does not need to contract peer agents in order to find out the most appropriate (optimal) offers for further problem delegation. Knowledge stored in the PLS will help the agent to decide by himself. It is obvious that limiting the communication traffic among

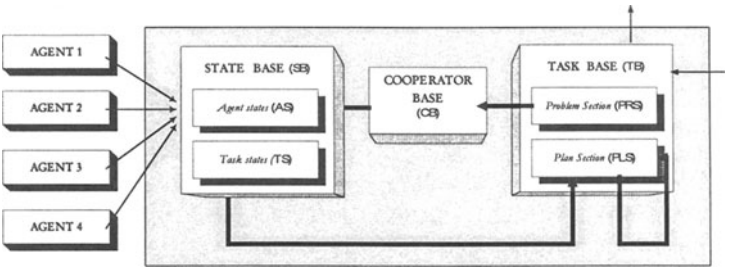

Figure 2 – Tri-base Model Planning

agents will in its own way decrease the computational complexity of the entire problem. The price we have to pay for this, is a communication increase among agents when updating the SB.

The model maintenance algorithms are based on a simple subscribe/advertise mechanism. After parsing the PrS knowledge, each agent identifies possible collaborators and subscribes these for reporting on their statuses. The subscribed agent advertises its load, capabilities, task completion times and cost estimates either

periodically or when either of these changes. This mechanism facilitates the agent to make the best decision with no further communication.

The communication savings of this approach may be seen as a specific communication load shift. The communication load is minimized in the agent's critical time (i.e.: in a moment when the agents are required to fulfill, through delegation, a request) while a new advertising activity appears in the agent's idle time. Moreover, each request is answered quickly but it brings substantial communication flows following the request fulfillment. The truth is that successful operation of such a multi-agent system depends on the community lifecycle. In order to utilize this acquaintance model based mechanism and to guarantee its communication savings for the frequency f of requirements to plan the following condition should be met

$$1/f \geq t_c + t_i,$$

where t_c is the maximum amount of time spent in the *critical time* planning and t_i is the maximum amount of time needed for processing the subscription/advertise mechanism in the agents' *idle time.*

There are two issues to be addressed here: (i) how many peer agents to subscribe and (ii) how many plans to keep pre-prepared. Cooperation neighborhood $\varepsilon^t(A)$ denotes a collection of such agents B who belong to agent's A scope of collaboration. All agents $B \in \varepsilon^t(A)$ are subscribed by the agent A at the time instant t. The maximum neighborhood $\varepsilon^t(A)$ is specified when the agent parses its problem solving knowledge-base (PRS) and detects who will be needed for further collaboration. The $\varepsilon^t(A)$ is fixed here for the entire course of agent's decision making. In this way, the agent is collecting lots of redundant data, which slows the community down substantially. The possible role of the meta-agent can be in analyzing the inter-agent communication and optimizing the $\varepsilon^t(A)$ neighborhood.

The cooperation neighborhood $\varepsilon^t(A)$ of a faulty agent A represents exactly its *subordinate zone* introduced in the definition of the afflicted zone in the context of re-planning. The complementary concept of the A's *collaborative zone* $\rho^t(A)$ includes all the agents which have subscribed the agent A for reports about his status. This set of agents can be described as follows

$$\rho^t(A) = \{Z \in \Theta : A \in \varepsilon^t(Z)\}$$

The production agents PA are terminal ones as they do all their work by themselves. They have no "subordinated" agents and their cooperation neighborhood $\varepsilon^t(A)$ is thus empty. Taking this into account we can define the *afflicted zone* of an agent A in a uniform way for all types of agents: the afflicted zone of the agent A is ($\varepsilon^t(A) \cup \rho^t(A)$).

On the other hand, $\pi^t(A)$ neighborhood, agent's A scope of reasoning, specifies how many of plans from PRS will be kept instantiated and on-line evaluated in PLS of the TB. We distinguish among two marginal cases. If $\pi^t(A) = \varnothing$, there is no plan pre-prepared and planning in the critical time requires substantially more of computational time for constructing the PLS plans. If $\pi^t(A)$ is maximal, there is no time needed for constructing the PLS plans, but each minor change within community results in massive re-computation and re-evaluation of the PLS plans. Designers of the system rely on meta-agent machine learning capabilities in setting $\pi^t(A)$ agent's scope of reasoning.

5. AUTONOMOUS RE-PLANNING

As there is no central plan available, no particular agent responsible for re-planning a given project exists. This is why as much of re-planning and re-configuration has to be carried out by agents themselves by coping intelligently with unexpected situations – **autonomous re-planning.**

Many strategies for re-planning can be found [Yang 97][Russell 95][Drummond 90][Lažanský 92] e.g. it is possible to re-plan all the already fixed processes from the points that correspond to the current state of physical manufacturing. This strategy is usually far from the (sub)optimal one. More sophisticated approaches based on certain knowledge-based 'backtracking' procedures are aimed at re-planning of some earlier fixed plans or their part only. The backtracking is guided by specific criteria which evaluate whether the blocked production zone (broken machine, machine occupied by urgent top-priority manufacturing etc.) in the expected time interval of blockage is needed for manufacturing the given product. To speed-up the evaluation process, some additional look-ahead markers should be created and stored in the plan sections of the agent task-bases. The markers contain structures resulting from a thorough analysis of the re-planning requirements. Let us denote as a **victim plan** such a plan suspending of which allows releasing the required resources.

Each autonomous re-planning process consists of three phases:

- Detection of the need for re-planning
- Selecting certain plans (victims) to be destroyed to release resources
- Re-planning itself which is planning considering the released resources

Let us consider the stages of re-planning in the ProPlanT system:

Stage A: Detection of the need for re-planning

a) In the case of the request-driven-re-planning agents do not have any difficulties to detect re-planning requirements. The agent himself realizes its inability to fulfill the request with a high priority marker.
b) in the case of the community-driven-re-planning the need is either detected by the meta-agent or by some of the agents who recognize that some of their subordinated agents failed:
   ba) PA died/failed: This can be recognized by the corresponding PMA(s) easily (as there is no answer or no acceptable answer reaching the PMA(s) or the execution of a certain plan failed)
   bb) PMA died/failed: This should be recognized by both the subordinated PAs and by the super-ordinated PPAs (all these being contained in the afflicted zone). PPA asks the other PMA(s) for help, the PAs broadcast the information about their statuses to all the PMAs. One (or more) of the PMAs takes over the responsibility of the died "colleague".

Stage B: Selecting plans:

a) In the case of the request-driven-re-planning: The relevant plans which occupy the most needed resources are selected as the ideal victims. Specific evaluation criteria are needed to detect the candidates for being victims.
b) In the case of the community-driven-re-planning all the prepared plans which are in the afflicted zone (all the plans which cannot be accomplished as the

corresponding PMA(s) or PA(s) died/failed) should be destroyed. The plans which were already partially carried out should be destroyed starting from the given stage of manufacturing.

Stage C: Planning using released resources

The planning of the missing plans (the re-planning itself) is started. This process runs in the same way for all the kinds of situations in the stages 1 and 2.

Let's mention that the tri-base acquaintance models enable to carry out all the three stages mentioned above in a very simple way. To destroy the *victim zone* means just to cancel the corresponding plans in the PLS. To re-plan means to create new items for the PLS using the knowledge stored in the PRS.

The critical point of the re-planning process is the strategy for finding the most suitable victim. This is not an easy task. Bad choice of a victim will generate huge amount of re-planning activities or extremely expensive delay in production. Therefore, the markers have to represent some heuristic value for the best victim. We have to find certain metric to be able to measure the complexity of the re-planning for a particular task. The complexity of the space, which has to be replanned, depends on the number of causal relations generated by such tasks and these relations are known during the plan construction. So we choose that number as a heuristic evaluation for re-planning. There are, of course, many other strongly problem dependent conditions, which have to be checked.

6. SUPERVISED RE-PLANNING

However, in many situations the community conflict cannot be detected on the peer-to-peer basis. If a collaborating agent fails to reply it may mean that it is dead or that is only overloaded (i.e. it cannot accept any other task but it will fulfill what has been already planned). An ideal member of the community who may be responsible for detecting this type of conflict is a meta-agent.

The meta-agent is an agent who is capable of some sort of meta-level reasoning about the multi-agent community. Unlike facilitators or brokers in classical agent based systems it is not central to the rest of agents. It neither controls the community nor serves as a communication center in the physical and symbolic sense. Instead, the meta-agent observes the communication traffic of the community and tries to draw corresponding conclusions regarding the agents' behavior with intention to improve performance of the whole system. By introducing this concept of the meta-agent we have tried to avoid making the community too fragile by relying on a single agent which becomes a community bottleneck otherwise.

The meta-agent monitors the behavior of the agents and induces knowledge about the community. Knowledge the meta-agent elicits may be further directly transmitted within the community in order to revise agents' local knowledge bases so that more efficient decision-making (or decision making considering deduced pieces of knowledge) could be performed. A number of revision types are considered in [Štěpánková 96]. Who identifies these changes and how they influence knowledge bases of the remaining agents? We can distinguish at least two reasons to trigger certain changes in the knowledge bases of any agent *C*. The agent *C* either identifies

something important in his environment (among his co-operators) or the need for updated is identified by the meta-agent.

In the former case, the revision processes to be described appear within the wrapper of the agent C only. If the need for change is identified by the meta-agent, the meta-agent has to define the set of agents concerned - often it can contain all the members of the community. After a detailed analysis we have found the following seven (three additional compared to [Cao 96, 97]) natural revision types as appropriate.

Let us describe in detail all the relevant revisions which any specific agent C has to do within its wrapper as soon as C identifies the need for revision (himself or due to information from the meta-agent). The same types of revisions have to be done simultaneously within the wrappers of all the agents if the source of impact for revision is the meta-agent, of course.

Revision 1: The agent A died and $A \in \alpha(C)$. The content of $\alpha(C)$ has to be revised for all the future activities. Thus $\alpha(C) \leftarrow \alpha(C) - \{A\}$ and $\varepsilon^t(C) \leftarrow \varepsilon^t(C) - \{A\}$ and this change has to be reflected in CB(C), the **cooperator base** of the agent C and in AS(C) (agent section of the **state base** of the agent C), from which the corresponding record has to be retracted. If A appears in a certain plan in the PLS(C) (plan section of the TASK BASE) this plan has to be retracted as well. Moreover, if PLS(C) contains no other solution for the supertask represented by the retracted plan, an alternative solution has to be suggested. If the agent C is not able to decompose this supertask any longer, this fact shall be advertised within agent's C subscribers within its *collaborative zone* $\rho^t(C)$. If the retracted plan is used as a part of the already existing global plan distributed within the community, a more elaborated process of re-planning has to be applied.

Revision 2a: The agent A stops being able to solve (coordinate) a task T (the agent didn't die but remains capable to solve some other tasks), i.e. $\beta(A) \leftarrow \beta(A) - \{T\}$ and $\omega(A,T) \leftarrow \varnothing$. The CB($C$) and TB($C$) have to be revised. Formally each CB(C) record starting with the agent A has to apply the new value of $\beta(A)$. Each record $\langle S,$ Dec(S), State(S), Trust(S)$\rangle \in$ TS(C), such that $\langle T,A\rangle$ appears Dec(S) and T has not been finished yet, shall be retracted and S re-planned. In a special case when $C = A$, each record in PRS(A) and PLS(A) records of the task T have to be retracted. Moreover, $\beta(A)$ and $\omega(A,T)$ have to be updated there. This revision shall be advertised to all subscribers of the agent A.

Revision 2b: The agent A stops being capable of decomposing a task T the way other agents expect him to do, i.e.: $\omega(A,T)$ is circumscribed. If there are other ways how the agent A can decompose the task T, the only information that has to be updated in the bases of collaborating agents is AS(C). The couple $\langle T,$ Cost(T)$\rangle \in$ Cap(A) has to be replaced with $\langle T,$ Cost'(T)$\rangle$ where Cost'(T) is a cost of another decomposition, the agent A can provide. If there is no other way how the agent A can decompose the task T, the problem reduces into the revision 2a.

Revision 3: A new agent A is born. To the CB(C) will be thus appended a new record $\langle A,$ Addr(A), Lang(A), $\beta(A)\rangle$. Functionality of this revision has been attributed to a special registering agent, who each newcomer registers with and who is in charge of distributing the contents of the CB.

Revision 4a: An existing agent A gained a new capability to solve/co-ordinate a new task T, e.g. by learning. Thus $\beta(A) \leftarrow \beta(A) \cup \{T\}$ and thus $\omega(A,T) \neq 0$. The $CB(C)$ as well as $TB(A)$ have to be revised according to this change.
Revision 4b: An existing agent A gained an alternative capability to solve/co-ordinate task a T (a new, more efficient decomposition has been identified) and $\omega(A,T)$ is extended in a corresponding way. There is no need to revise $CB(C)$ in this case, however this revision will affect the $AS(C)$ and $PRS(C)$. Each record $\langle A, Cap(A), Load(A), Trust(A)\rangle \in TS(C)$ such that $\langle T, Cost(T)\rangle \in Cap(A)$ shall be replaced with a record $\langle A, Cap'(A), Load(A), Trust(A)\rangle$ where $Cap'(A) = Cap(A) - \{\langle T, Cost(T)\rangle\} \cup \{\langle T, Cost'(T)\rangle\}$, where $Cost'(T)$ is the cost of decomposing the task T using the new decomposition. Revising the $AS(C)$ will trigger re-computation of the $PLS(C)$.
Revision 5: The Agent A changes some of his properties, e.g. his load estimate $Load(A)$ or trust value $Trust(A)$ as a result of expected problem solving activities or increasing/decreasing computational power of the agent A. This revision affects $AS(C)$ of the collaborating agents to which this shall be broadcast. If the agent's change results in decreasing capability to cope with requests, this revision reduces into revision 2 with all its consequences similarly to possible re-planning.
Revision 6: The Agent A changes formulation of his applicability constraint C' conditions for specific decomposition of a task T. If the re-specification of the constraint is a general property of the task T (detected by the meta-agent), it affects $TB(C)$ of all the agents C for which $T \in \beta(C)$. Firstly, $PRS(C)$ has to be updated by re-placing the original task decomposition of T with a new plan with an updated set C'. $PLS(C)$ has to be inspected and re-computed accordingly.

Bases of the agents are revised twofold. Firstly, agents revise their bases autonomously by **subscribe/advertise** mechanism. Apart from this, the meta-agent periodically carries out *capability* revisions in some system idle time. Consequently, the bases contain all the time up-to-date or nearly up-to-date information.

7. EXPERIMENTS AND EXPERIENCE

The re-planning algorithms have been designed, implemented and tested within the ProPlanT project on the community of agents planning and simulating production of TV/FM transmitters at Tesla-TV manufacturing enterprise. The first experiments on the test case data confirmed that re-planning is highly desirable especially in the case of project-oriented production, where single, unique products are manufactured. As a matter of fact, the proposed re-planning approach was found extremely robust in the case of this type of manufacturing: The failure of a PA agent resulted in an immediate detection of the afflicted zone by a simple checking of the contents of the PLSs on the PMA level. Each failure of a PMA agent required – as a rule – somewhat more complicated search on both the PA and PMA levels. The supervised re-planning is also very simple. But, the multi-agent system structure seems to be – in the case of production planning for project driven manufacturing – fixed and stable enough to carry out both the autonomous and supervised re-planning without any need of a meta-agent advise. Our further experiments shown that in the case of a more flexible and changeable MAS structures, like in the systems where dynamic coalitions are permanently created or in the geographically distributed MAS-based

communication/information systems, the needs for meta-agents' interference are crucial.

8. CONCLUSIONS

We have shown that the problem of re-configuration and re-planning in multi-agent systems is not an easy problem and needs to be well analyzed. According to our experience, the tri-base acquaintance model, stored in the communication wrapper of each agent, suits very well flexibility and dynamics required when planning and re-planning complex manufacturing processes. We have introduced an alternative re-planning solution exploiting fully the role of a meta-agent, an agent observing the functionality of the community. The advantages of the re-planning in the multi-agent community exploring the tri-base acquaintance models of behavior were documented: The mutual agents' awareness, well-defined, but modest–in-volume communication and clearly granulized knowledge in the tri-base models represent features which make the re-planning process much simpler. The only problem, which seems to be crucial is the strategy of selecting the "ideal" victim plans to be destroyed before a new step of re-planning.

This research was supported by the EC funded project EUREKA No. 1439 and the Czech Ministry of Education, Youth and Sport (MSMT) grants No. VS 96047 and J04/98:212300013.

REFERENCES

[Cao 97] Cao W., Bian C.-G., Hartvigsen G.: Achieving Efficient Cooperation in a Multi-Agent System: The Twin-Base Modeling. In: *Co-operative Information Agents.* LNAI 1202, Springer-Verlag, Heidelberg, 1997, pp. 210-221

[Drummond 90] Drummond M., Tate A.: AI Planning, In: *Knowledge Engineering Fundamentals* (Hojjat Acheli ed.), Mc Graw Hill Publishing Company, 1990

[Lazansky 92] Lazansky J.: Practical Applications of Planning Tasks. In*: Advanced Topics in Artificial Intelligence.* LNAI No. 617, Springer Verlag, Heidelberg, 1992

[Russell 95] Russell S., Norwig P.: *Artificial Intelligence – A modern Approach*, Prentice Hall Publishing Company, New Jersey 1995

[Pěchouček 00] Pěchouček M., Mařík V., Štěpánková O.: "Role of Acquaintance Model in Agent-Based Production Planning in Workshop of Cooprative Information Agents, Boston 2000

[Štěpánková 98] Štěpánková O., Pěchouček M., Mařík V., Lažanský J.: Applications of Model-Based Co-Operative Agents. In: Production Planning In: *Database and Expert Systems Applications*, LNCS No.1460, Springer Verlag, Heidelberg, 1998,

[Yang 97] Yang Q.: Intelligent Planning: A Decompostion and Abstraction Based Approach, Springer and Verlag, Heidelberg, 1997.