

APPLICATION OF DIAKOPTICAL MAS FRAMEWORK TO PLANNING PROCESS MODELLING

Borue, S. U., Ermolayev, V. A. Tolok, V. A.

Dept. of Mathematical Modelling and Information Technologies,

Zaporozhye State University,

66, Zhukovskogo st., 330600, Zaporozhye, Ukraine, tel/fax:+380 61 264 17 24,

E-mail: {bsu, eva, tolok}@zsu.zaporizhzhе.ua

Abstract. The paper presents the case study of the application of the theoretical framework* of modelling the processes of information interchange among the members of evolving intellectual agent communities to modelling of one of practically important types of business processes - a planning process. A business process in frame of the presented research is denoted as a set (task) of atomic works. In the frame of the applied modelling approach the members of dynamically formed, scalable and evolving intellectual agent communities act as functional components performing the atomic works of such a task. The principal alterity of the approach is the usage of parametric feedbacks and, alternatively, agent state constraints to atomic works execution sequence control. Case study analysis shows that the framework is practically applicable to modelling of planning processes. Furthermore, the hypothesis that the framework is applicable to another types of business processes is discussed.

Аннотация: В статье рассматривается применение теоретического аппарата* моделирования процессов информационного обмена в сообществах интеллектуальных агентов к моделированию одного из практически важных типов бизнес процессов - процесса планирования. В рамках представляемых исследований бизнес-процесс определяется как множество (задание), состоящее из атомарных работ. Функциональными компонентами, выполняющими атомарные работы такого задания, в примененном подходе к моделированию являются члены динамически формирующихся, масштабируемых и эволюционирующих сообществ агентов. Принципиальным отличием данного подхода является применение параметрических обратных связей и, альтернативно, ограничений состояний агентов для управления последовательностью выполнения атомарных работ. Анализ представленного приложения показывает, что данный аппарат практически применим к моделированию процессов планирования. Более того, выдвигается гипотеза о его применимости к другим типам бизнес процессов.

Keywords: Business Processes, Intelligent Agent, Evolving Agents Community, Scalability.

Ключевые слова: Бизнес процесс, интеллектуальный агент, эволюционирующее сообщество агентов, масштабируемость.

The paper is submitted to UkrPROG'2000, May 23-26, 2000, Kiev, Ukraine.

* The research presented is run in frame of the Project financed by Ukrainian Ministry of Education, Grant № 0199Y1571. Intermediate results were reported and discussed at ESPIRIT AgentLink I2A SIG meetings.

APPLICATION OF DIAKOPTICAL MAS FRAMEWORK TO PLANNING PROCESS MODELLING

Borue, S. U., Ermolayev, V. A. Tolok, V. A.

Dept. of Mathematical Modelling and Information Technologies,
Zaporozhye State University,
66, Zhukovskogo st., 330600, Zaporozhye, Ukraine, tel/fax:+380 61 264 17 24,
E-mail: {bsu, eva, tolok}@zsu.zaporizhzhе.ua

Abstract. The paper presents the case study of the application of the theoretical framework* to modelling of one of practically important types of business processes - a planning process. A business process in frame of the presented research is denoted as a set (task) of atomic works. The diakoptical framework applied models business processes as the processes of information interchange among the members of dynamically formed, scalable and evolving intellectual agent communities. The principal aliterity of the approach is the usage of parametric feedbacks and, alternatively, agent state constraints to atomic works execution sequence control. Case study analysis shows that the framework is practically applicable to modelling of planning processes. Furthermore, the hypothesis that the framework is applicable to another types of business processes is discussed.

Аннотация: В статье рассматривается применение теоретического аппарата* моделирования процессов информационного обмена в сообществах интеллектуальных агентов к моделированию одного из практически важных типов бизнес процессов - процесса планирования. В рамках представляемых исследований бизнес-процесс определяется как множество (задание), состоящее из атомарных работ. Функциональными компонентами, выполняющими атомарные работы такого задания, в примененном подходе к моделированию являются члены динамически формирующихся, масштабируемых и эволюционирующих сообществ агентов. Принципиальным отличием данного подхода является применение параметрических обратных связей и, альтернативно, ограничений состояний агентов для управления последовательностью выполнения атомарных работ. Анализ представленного приложения показывает, что данный аппарат практически применим к моделированию процессов планирования. Более того, выдвигается гипотеза о его применимости к другим типам бизнес процессов.

Introduction

The models of the processes of information interchange in distributed intelligent systems are often based on the usage of structured collections of rigid relationships among the participating functional nodes. These representations are inherited from the known organisational and information models and unfortunately are too much static and do not fully cover the things we humans deal with in real life. If we take decision making process for an instance, it can not be adequately modelled by, say, a hierarchy of rigidly positioned actors. Humans apply much more soft interrelationships to solve the problem: brainstorming, informal discussion, negotiation, whatever. Another big pitfall is that rigid relationship models are not really scalable. The demand for modelling soft temporary relationships (like associations, discussions in real life) in scalable systems is emerging in diverse areas: intelligent information integration, enterprise modelling, business process modelling, simulation, etc. The solutions today are mostly represented by the frameworks, architectures and implementations exploiting the paradigms of an Intellectual Agent and a Multi Agent System (MAS). Good examples from various application domains are [1-3].

The authors have proposed [4] the framework for modelling of the processes of functional interrelationship among the distributed components (presented by intellectual agents) of a dynamic multifunctional enterprise/department information system focusing on the fact that the associations between participating nodes/actors are soft and are changing in time. The approach is based upon the principle of diakoptics [5] and macromodelling methodology [6]. Diakoptics, as understood in this work, is the equilibrium upon the following a bit tricky things:

- Representation of system components as well as the system as a whole by means of the functionally equivalent, unified and simplified software model (an intellectual agent with its behaviour models presented by macromodel programs)
- Embedment of components' interrelationship topology into component macromodel
- Embedment of components' specialisation into component macromodel

The advantages of the framework are seen as follows: there is the possibility to apply generic models to presenting the components, their intercommunication, subsystems and a system as a whole; there are good opportunities to operate

* The research presented is run in frame of the Project financed by Ukrainian Ministry of Education, Grant № 0199Y1571. Intermediate results were reported and discussed at ESPIRIT AgentLink I2A SIG meetings.

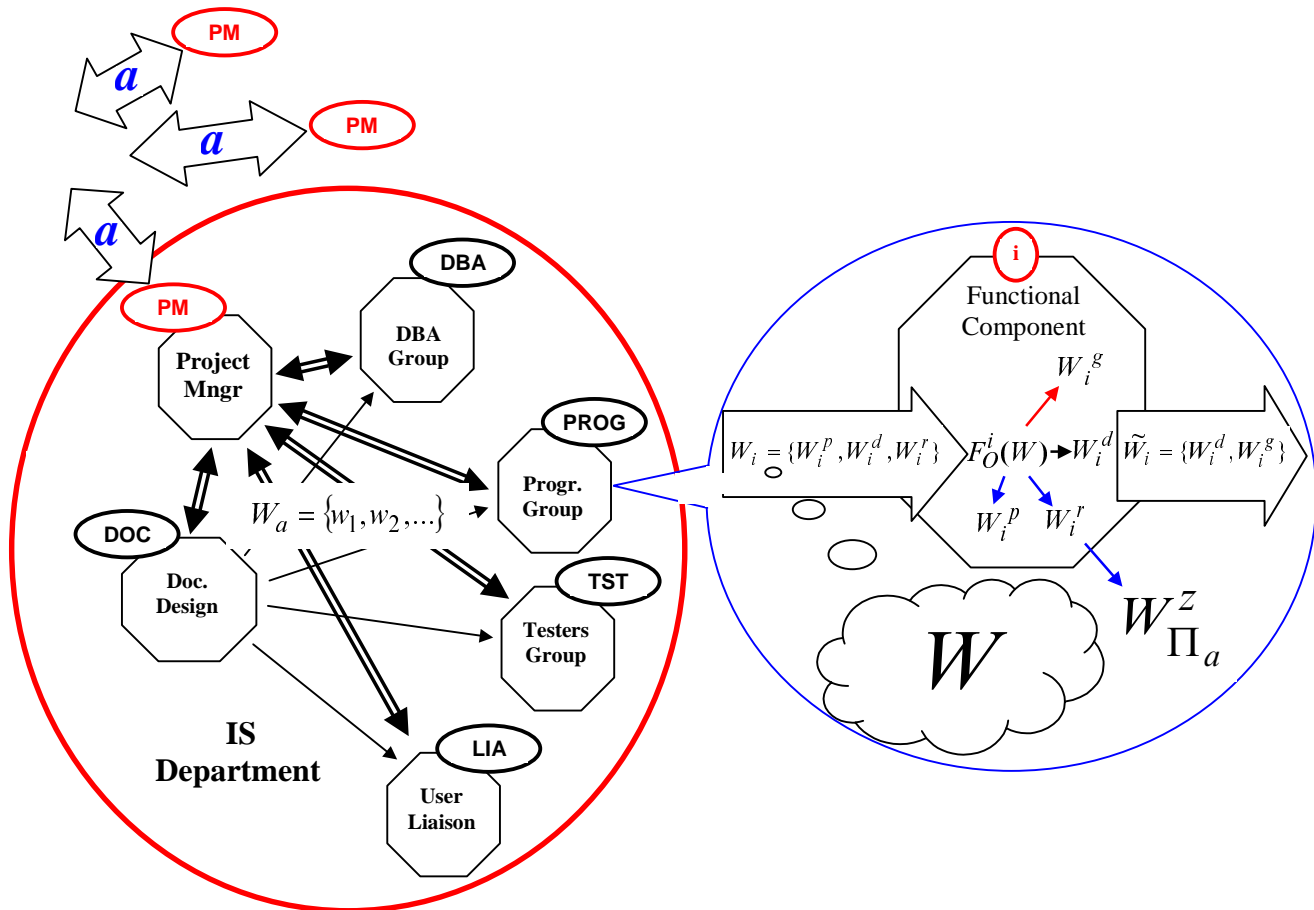


Fig. 1. Functional system/component model.

with subsystems by means of much more simple component models irrespectively of the nest levels inside; there are the capabilities to use generic architectures, interfaces and finally implement well scalable applications. Instead of creating pretty complex, rigid models with tightly linked components, the preference of the method is to operate with collections of 'autonomous' agents dynamically forming communities for one or another process of information interchange (a task). This reduces complexity, adds movement and is quite appropriate to model business processes as the processes of information interchange by solving the problem by part.

In this paper we leave aside in-depth discussion of semantic aspects (ontology representation, interoperability, etc.) as well as architectural and implementation issues leaving this peace of the cake for future publications. We concentrate on the verification of the applicability of the framework to modelling of one of practically important types of business processes - a planning process.

The paper is structured as follows. Section 1 briefly presents the framework used in planning process case study. Section 2 provides the case study. The process of an Information System (IS) development proposal preparation is modelled by simulation. Participating agents dynamically form the community for the task, which is consequently solved by part. After the task is accomplished the results are used to generate the plan for the real performance. Section 3 provides the analysis of the case study. Finally the results are summarised and the directions of the future work are presented.

1. Modelling Framework

The main idea used in the framework is that a business process is modelled as the process of information interchange among the members of a dynamic community of intellectual functional actors - a functional system. The actors are modelled by intellectual information agents capable to communicate with each other by means of the defined set of communicative acts with parametric feedbacks [4]. These communicative acts comply with ACL [7] and KQML [8] capabilities. A task is assumed to be the set of atomic works. Each actor is capable to perform some atomic works from the set of permissible atomic works of the functional system.

The framework provides the generic model for a functional actor -- agents' model, the generic communication model for task-related actors' community -- agents' community model and the generic model for the task execution -- process model.

At the agent level the framework uses a kind of BDI [9] model. It provides the key agent's characteristics [10] of situatedness, autonomy and flexibility. Its role is to receive external influences, to verify if the incoming influence complies with the agent's role and finally to adjust its behaviour and perform appropriate macromodel program -- i.e. execute or reject the atomic work requested by the input influence. The function of the macromodel is also to form the feedback containing the results. The results may be presented as functions from the parameters of the incoming influence. Formally (see [4] for details) the generic agent comprises its sensory interface, the cascade of 3 finite-state machines for incoming influence verification and the macromodel execution block.

At the community level it is assumed that the agents taking part in the process of task execution communicate by means of the following performatives:

- **Directive** - the routine to influence the counterpart to unconditionally execute the atomic work.
- **Determined request** - the routine to influence the counterpart to execute the atomic work and to request the results back.
- **Determined request with results analysis** - the routine to influence the counterpart to execute determined request and consequently to reason about the parametrical results received as the reaction.
- **Undetermined request with results analyses** - the routine to broadcast the influence in case the executor agent is unknown. The influence is multicasted to every agent within the community. The results are afterwards compared and appropriate reasoning is performed.

At the functional system level some basic assumptions are made to simplify the framework. Functional community members are assumed to be strongly oriented to teamwork and fair play. Successful task execution has higher priority than local goals of a certain agent. The agents joining the community are bounded to deliver truthful results even if it contradicts to their local goals.

Functional System/Component Model. The model of a functional system as well as a functional component model is built upon the idea of "absorption" and "generation" of atomic works from the set of permissible works $W = \{w_1, w_2, \dots\}$ -- refer to Fig. 1. It is considered that the sensory input of the functional component i admits a task $W_i \subseteq W$. A certain part of its works W_i^p may be performed ("absorbed") by the given component and the remaining part of works may be either redirected to another system's components W_i^d in case functional component knows the recipient(s), or rejected W_i^r . Functional component may as well generate additional set of works W_i^g to complete the execution of works W_i^p . W_i^g as well as W_i^d are redirected to another components:

$$W_i \rightarrow F_O^i(W) \rightarrow \tilde{W}_i, \quad (1a)$$

where: $W_i = \{W_i^p, W_i^d, W_i^r\}$, $\tilde{W}_i = \{W_i^d, W_i^g\}$, $F_O^i(W)$ - macromodel program.

In a special case component i may generate a new set of works W_i^g without been invoked by incoming influence W_i - i.e. may "summon" a new task:

$$F_O^i(W) \rightarrow \tilde{W}_i, \quad (1b)$$

where: $\tilde{W}_i = \{W_i^g\}$, $F_O^i(W)$ - macromodel program.

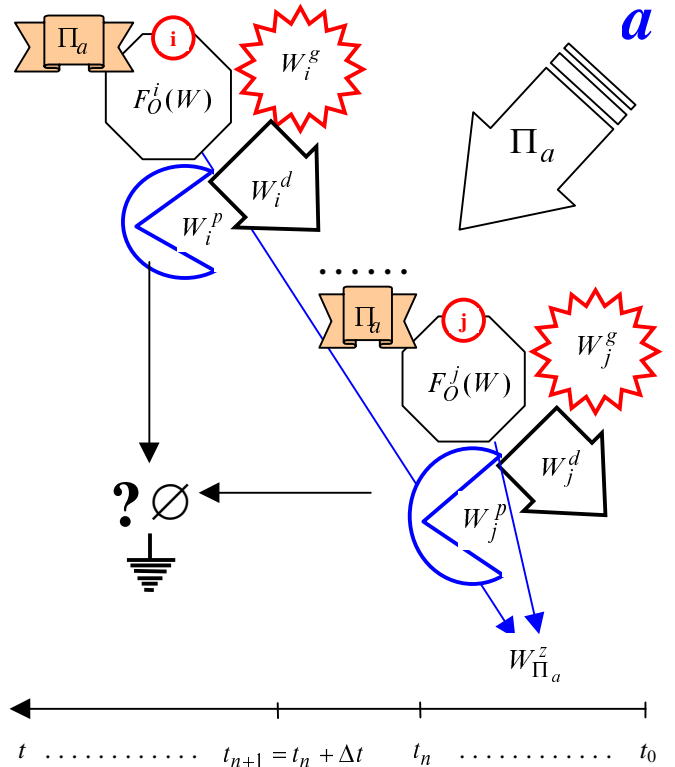


Fig.2. Process model.

Thus, the task of functional component/system model is appropriate execution of (1a) and (1b) - see Fig. 1. The model is constrained by the rule that the duration of execution of each atomic work $w_j \in W$ is the definite time interval Δt .

A functional system is tailored to perform processes. A process is denoted as the procedure of task execution. Process Π_a starts with generation of the new task $W_a \subseteq W$. Task W_a as well as the additional tasks \tilde{W}_a are considered to be linked to process Π_a and labelled with the unique identifier of this process. The component is considered to be **linked to process** Π_a in case it has absorbed the part of W_a , \tilde{W}_a , or has generated W_a^g .

Process Π_a is considered to be completed in case all the components stopped to absorb the atomic works of the tasks linked to process Π_a . The set of works $W_{\Pi_a}^z$ not absorbed in the process of Π_a is denoted as the set of **inexecutable** works. See the chart on Fig 2.

Modelling of process Π_a (steady-state mode) is performed by applying (1b) and (1a) to all of the components of the system until the process is completed.

Dependency $F_O^i(W)$ is modelled in frame of generic agent model [4], or by any other appropriate method. The requirement to this part of functional component model is the adequate execution of (1a), (1b).

For practice it seems to be reasonable to restrict the set of system's permissible atomic works and to consider it to be finite: $W = \{w_1, w_2, \dots, w_\sigma\}$. This restriction implies the possibility to apply macromodelling principle [6] to functional system modelling. Macromodelling provides the possibility to skip topological analyses of system components and assumes utilisation of the benefits of diakoptical approach. Modelling of the system as a whole while performing a task is herein organised as a two-level process performed sequentially at discrete time points $t_n, t_{n+1} = t_n + \Delta t$.

Task Execution Model. Let $W = \{w_1, w_2, \dots, w_\sigma\}$ be the set of permissible atomic works of the functional system.

At the first (upper) level the assembly of all the components' states into the conjoint system states model at the moment $t_n + \Delta t$ is performed. The conjoint model is presented in the form of matrix $\Omega(t_n + \Delta t)$ with dimension $m \times \sigma$, where m is the number of system components and σ is the number of atomic works in W . The rows of matrix Ω (see Fig. 3.) are the vectors $\Theta_i = \{k_1, k_2, \dots, k_j, \dots, k_\sigma\}$ reflecting components' states, where k_j is the state of the component i with respect to the execution of atomic work w_j . In the simplest case the role of parameter k_j is as follows:

$k_j = 0$ - the component is executing atomic work w_j ; $k_j = l > 0$ - the component is executing the work w_j and l similar works are waiting in line;

$k_j = l < 0$ - the component was capable but has not executed l atomic works w_j (idle state).

System states matrix $\Omega(t_n + \Delta t)$ is formed from the matrixes \mathbf{K}_i (dimension $m \times \sigma$) representing component states. Matrixes \mathbf{K}_i are produced by the executable $F_O^i(W)$ of the component model at the second modelling level to provide the inputs to following formula:

$$\Omega = \sum_{i=1}^n \mathbf{K}_i . \quad (2)$$

In addition work delays vector D_a of the process Π_a is updated in case one or more works w_j from W_i are redirected by any functional component to itself:

$$D_a[j] = D_a[j] + 1 \quad (3)$$

At the second (lower) level the production of \mathbf{K}_i is performed for each system component. The components, as mentioned before, are modelled arbitrarily ($F_O^i(W)$), but it is provided that the input information for component i is the vector Θ_i , the matrix $\Omega(t_n)$ and the matrix \mathbf{K}_i defined for the previous time point t_n . Matrix \mathbf{K}_i is built according to the rule presented at Fig. 4. and should adequately reflect the behaviour of the component within the time interval $[t_n, t_n + \Delta t]$,

where:

$k_{lj}, l \neq i$: **1** - component i allocates work w_j to component l ,
 0 - otherwise

- k_{ij} :
- 1 - component absorbs (or is capable to perform) work w_j within interval $]t_n, t_n + \Delta t]$,
 - 1 - component i allocates work w_j to itself,
 - 0 - component i is not capable to perform work w_j within given time interval $]t_n, t_n + \Delta t]$.

The analysis of $\Omega(t_n)$ values may thus provide to univocally evaluate component load, idle state share within each time interval $]t_n, t_n + \Delta t]$ and therefore to reason about the necessity of changing its behaviour $F_O^i(W)$ in the future.

2. Planning Process Case Study

One of the possible applications of the presented modelling framework is business process planning. The main idea is close to the one of ADEPT [1] project: modelling business processes as a collections of autonomous, problem solving agents which interact to model business process execution. Corresponding plan may thus be afterwards automatically generated.

Let's discuss how the framework may be practically applied to information system design project planning. We'll assume that the organisational structure, which may be assigned to the project implementation, is that shown on Fig. 1. -- IS Department. Other assumptions are:

- IS Department shown on Fig. 1. is one of many candidates to be selected for the project. From the point of view of the competition IS Department as well as the other parties are considered to be the member agents representing functional components of the higher level community.
- Agents community of IS Department is modelled in details and comprises Project Manager (**PM** agent), DBA Group (**DBA** agent), Programmers Group (**PROG** agent), Documentation Design Group (**DOC** agent), Testers Group (**TST** agent), User/Customer Liaison Group (**LIA** agent) as functional components.
- In case on some stage the necessity emerges to go in more details with one of the functional components listed, this component may be modelled as a functional system - agents community.

The process starts at the point t_0 when external influence

$$W_a = \{w_0 = \text{'Propose_IS_Development_Plan'}, X_0, Y_0\}$$

with the parameters and the result descriptions

$$X_0 = \{\mathbf{budget} = \langle \text{figure} \rangle, \mathbf{duration} = \langle \text{figure} \rangle, \mathbf{Proposal_Template} = \langle \text{file_name} \rangle,$$

$$\mathbf{Proposal_Descr} = \langle \text{file_name} \rangle\};$$

$$Y_0 = \{\text{Possibility}(\mathbf{budget}, \mathbf{duration}), \text{Proposal}(\mathbf{Proposal_Template})\}$$

comes to **PM**'s sensory input. **PM** performs at least the following actions:

1. **Input influence verification.** Within this step **PM** routine is to check out if the input influence, its parameters and result descriptions comply with **PM**'s role and current state constraints. The further behaviour of **PM** (i.e. macromodel program) is chosen from two alternatives: the influence is accepted or the influence is rejected. For brevity rejection macromodels will not be discussed in this case study. We'll further consider that all the input influences are accepted and appropriate macromodel programs are executed.
2. **Input work execution and agent state change.** Within this step **PM**'s macromodel $F_O^{PM}(W_a)$ (in accordance with software project planning ontology provided by the community ontology agent) decomposes incoming work - i.e. generates the set of works corresponding to the project stages following the rule (1a) with:

$$W_{PM}^p = \{w_0 = \text{'Propose_IS_Development_Plan'}, X_0, Y_0\}, W_{PM}^d = \emptyset, W_{PM}^r = \emptyset,$$

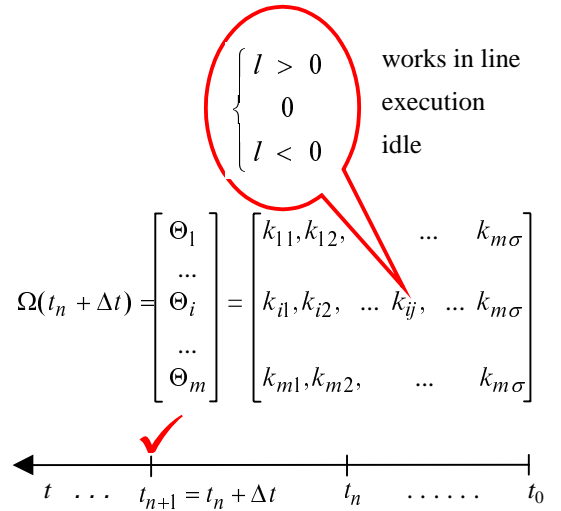


Fig. 3. System state at $t_n + \Delta t$

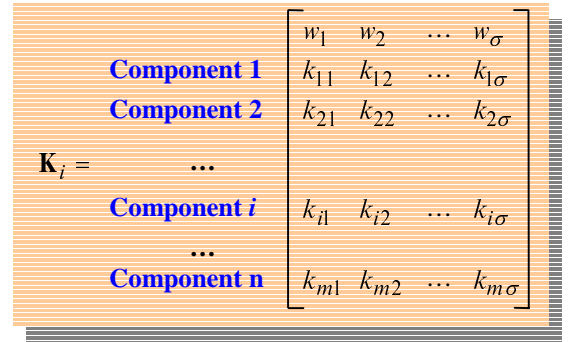


Fig. 4. Component capabilities and intentions

$$\begin{aligned}
W_{PM}^g = & \{ w_1 = ('Assemble Project Proposal', X_1, Y_1), \\
& w_2 = ('Choose best DB Schemata Plan Bid', X_2, Y_2), \\
& w_3 = ('Choose best Software Model Plan Bid', X_3, Y_3), \\
& w_4 = ('Choose best REQ Analyses Plan Bid', X_4, Y_4), \\
& w_5 = ('Analyse requirements', X_5, Y_5), \\
& w_6 = ('Design database schemata', X_6, Y_6), \\
& w_7 = ('Design software model', X_7, Y_7), \\
& w_8 = ('Program the software', X_8, Y_8), \\
& \dots\dots\dots \\
& w_{15} = ('Perform customers training ', X_{15}, Y_{15}) \}.
\end{aligned}$$

At this step of macromodel program execution the change of **PM**'s state s_0 to the state s_1 by adding additional state constraints may be performed in case it is requested by the behaviour rule provided by appropriate ontology or is encapsulated within the macromodel program.

3. **Component matrix $K_{PM}(t_0)$ generation.** Within this step **PM** macromodel generates $K_{PM}(t_0)$ as shown on Fig. 5.

	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	...	w_{15}
DBA	0	0	0	0	1	1	1	0		0
PROG	0	0	0	0	0	1	1	1		0
PM	1	1	1	1	0	0	0	0	...	0
DOC	0	0	0	0	0	0	0	0		0
TST	0	0	0	0	0	0	0	0		0
LIA	0	0	0	0	1	1	1	0		1

Fig. 5. **PM** capabilities and intentions for the task $W_a = \{ 'developIS', X, Y \}$ within $[t_0, t_0 + \Delta t]$.

	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	...	w_{15}
Θ_{DBA}	0	0	0	0	1	1	1	0		0
Θ_{PROG}	0	0	0	0	0	1	1	0		0
Θ_{PM}	1	1	1	1	0	0	0	0	...	0
Θ_{DOC}	0	0	0	0	0	0	0	1		0
Θ_{TST}	0	0	0	0	0	0	0	0		0
Θ_{LIA}	0	0	0	0	1	1	1	0		1

Fig. 6. System state (related to W_a) at $t_0 + \Delta t$.

$K_{PM}(t_0)$ may be interpreted as follows: for the next modelling step $t_0 + \Delta t$ works w_1, w_2, w_3, w_4 are planned for self-performance; works w_5, w_6, w_7 are redirected to more than one functional component: w_5 - to **DBA** and **LIA** agents, w_6 - to **DBA**, **PROG** and **LIA** agents, w_7 - to **DBA**, **PROG** and **LIA** agents. The reason for these multiply redirections is the attempt to obtain optimal task performance by choosing the component with the best bid. Another reason may be that **PM** doesn't really know well what are the duties of **DBA**, **PROG**, **LIA** and wants to gain some experience for the future. We do not however consider that **PM** may decide to assign works to multiply counterparts for partial execution as far as this assumption contradict to work atomicity. Works w_5, w_{15} are assigned explicitly to agents **PROG** and **LIA**.

System state matrix $\Omega_a(t_0 + \Delta t)$ related to the process $W_a = \{ 'Propose_IS_Development_Plan', X, Y \}$ will obviously look like shown on Fig. 6. as all agents except **PM** are idle within $t_0 + \Delta t$.

Let's examine the activities of **PM**, **DBA** and **LIA** agents at the next modelling step $t_1 = t_0 + \Delta t$.

PM accepts the set of works

$$\begin{aligned}
W_{PM} = & \{ w_1 = ('Assemble project proposal', X_1, Y_1), \\
& w_2 = ('Choose best DB Schemata Plan Bid', X_2, Y_2), \}, \\
& w_3 = ('Choose best Software Model Plan Bid', X_3, Y_3), \\
& w_4 = ('Choose best REQ Analyses Plan Bid', X_4, Y_4) \}
\end{aligned}$$

with the parameters and result descriptions for w_1 :

$$\begin{aligned}
X_1 = & \{ \mathbf{budget} = \langle \text{figure} \rangle, \mathbf{duration} = \langle \text{figure} \rangle, \\
& \mathbf{Proposal_Template} = \langle \text{file_name} \rangle, \\
& \mathbf{Proposal_Descr} = \langle \text{file_name} \rangle, \\
& \tilde{Y}_2, \tilde{Y}_3, \tilde{Y}_4, \tilde{Y}_8, \dots, \tilde{Y}_{15} \}, \\
Y_1 = & \{ \mathbf{Possibility}(\mathbf{budget}, \mathbf{duration}), \mathbf{Proposal}(\mathbf{Proposal_Template}) \}
\end{aligned}$$

As far as at t_1 the parameters $\tilde{Y}_2, \tilde{Y}_3, \tilde{Y}_4, \tilde{Y}_8, \dots, \tilde{Y}_{15}$ do not yet contain valid values, **PM**'s finite-state machine F_X resolves into rejection state and work w_1 is classified as belonging to the set of redirected works W_{PM}^d . Thus,

work w_1 is actually delayed at least for the period of Δt and **PM** macromodel adds 1 to element d_1 of the process delays vector D_a . Work w_1 is again scheduled to **PM** for the next modelling step t_2 . The same is true for works w_2 , w_3 , w_4 as well.

DBA accepts the set of works $W_{DBA} = \{ w_5 = ('Analyse\ requirements', X_5, Y_5), w_6 = ('Design\ database\ schemata', X_6, Y_6), w_7 = ('Design\ software\ model', X_7, Y_7) \}$ with corresponding parameters and result descriptions:

$$X_5 = \{ x_5^1 = (\mathit{budget} = \langle b \rangle), x_5^2 = (\mathit{duration} = \langle d \rangle), \\ x_5^3 = (\mathit{Plan_Template} = \langle \mathit{file_name} \rangle), x_5^4 = (\mathit{Step_Descr} = \langle \mathit{file_name} \rangle) \},$$

$$Y_5 = \{ y_5^1 = (\mathit{Possibility}(\mathit{budget}, \mathit{duration})), \\ y_5^2 = (\mathit{Plan_File_Name}(\mathit{Plan_Template})) \};$$

$$X_6 = \{ x_6^1 = (\mathit{budget} = \langle b \rangle), x_6^2 = (\mathit{duration} = \langle d \rangle), \\ x_6^3 = (\mathit{Plan_Template} = \langle \mathit{file_name} \rangle), x_6^4 = (\mathit{Step_Descr} = \langle \mathit{file_name} \rangle) \},$$

$$Y_6 = \{ y_6^1 = (\mathit{Possibility}(\mathit{budget}, \mathit{duration})), \\ y_6^2 = (\mathit{Plan_File_Name}(\mathit{Plan_Template})) \};$$

$$X_7 = \{ x_7^1 = (\mathit{budget} = \langle b \rangle), x_7^2 = (\mathit{duration} = \langle d \rangle), \\ x_7^3 = (\mathit{Plan_Template} = \langle \mathit{file_name} \rangle), x_7^4 = (\mathit{Step_Descr} = \langle \mathit{file_name} \rangle) \},$$

$$Y_7 = \{ y_7^1 = (\mathit{Possibility}(\mathit{budget}, \mathit{duration})), \\ y_7^2 = (\mathit{Plan_File_Name}(\mathit{Plan_Template})) \};$$

where: $\mathit{Possibility}(\mathit{budget}, \mathit{duration})$ is the result description presented on Fig. 7.; $\mathit{Plan_File_Name}(\mathit{Plan_Template})$ is estimated to be either returned as string "DUMMY" in case the plan has not been produced, or as string containing the valid file name produced according to $\mathit{Plan_Template}$.

The processing of 'Analyse requirements' policy by **DBA**'s finite state machine F_A leads to the rejection of this policy as it is considered in our case study to be unauthorised for **DBA**. Rejection policy executor forms results vector $\tilde{Y}_5 = \{\mathbf{O}, "DUMMY"\}$. The same reaction will be produced while processing 'Design software model' work: $\tilde{Y}_7 = \{\mathbf{O}, "DUMMY"\}$. Where \mathbf{O} is the matrix produced according to the rule given on Fig. 7. and containing zero possibility values: $\mathbf{O}_{ij} = 0, i = 1, \dots, 5, j = 1, \dots, 5$. 'Design database schemata' work is performed by **DBA** and the following result is returned:

$$\tilde{Y}_6 = \left\{ \begin{bmatrix} 0.1 & 0.1 & 0.4 & 0.6 & 0.35 \\ 0.3 & 0.35 & 0.4 & 0.8 & 0.35 \\ 0.3 & 1.0 & 1.0 & 0.8 & 0.35 \\ 0.35 & 1.0 & 1.0 & 0.85 & 0.35 \\ 0.4 & 1.0 & 1.0 & 0.9 & 0.35 \end{bmatrix}, "DBSCH_PLAN.XLS" \right\}, \quad (4)$$

LIA agent accepts the same as **DBA** set of works $W_{LIA} = \{ w_5 = ('Analyse\ requirements', X_5, Y_5), w_6 = ('Design\ database\ schemata', X_6, Y_6), w_7 = ('Design\ software\ model', X_7, Y_7) \}$. 'Analyse requirements' work is performed by **LIA** and the following result is returned:

Θ_{DBA}	0	0	0	0	0	0	0	0	0	DBA	0	0	0	0	1	0	1	0	0
Θ_{PROG}	0	0	0	0	0	0	0	0	0	PROG	0	0	0	0	1	1	0	0	0
$\Omega_a = \Theta_{PM}$	1	1	1	1	0	0	0	0	...	$W_{\Pi_a}^z = PM$	0	0	0	0	0	0	0	0	...
Θ_{DOC}	0	0	0	0	0	0	0	0	0	DOC	0	0	0	0	0	0	0	0	0
Θ_{TST}	0	0	0	0	0	0	0	0	0	TST	0	0	0	0	0	0	0	0	0
Θ_{LIA}	0	0	0	0	0	0	0	0	0	LIA	0	0	0	0	0	1	1	0	0

Fig. 8. System state and inexecutable works (related to W_a) at $t_1 + \Delta t$.

	0.5d	0.8d	1.0d	1.5d	1.8d
0.5b	p	p	p	p	p
0.8b	p	p	p	p	p
1.0b	p	p	p	p	p
1.5b	p	p	p	p	p
1.8b	p	p	p	p	p

Possibility: estimated values $p \in [0,1]$

Fig. 7. Result description example.

$$\tilde{Y}_5 = \begin{bmatrix} 0.1 & 0.15 & 0.4 & 0.6 & 0.55 \\ 0.1 & 0.55 & 0.4 & 1.0 & 0.65 \\ 0.2 & 0.9 & 1.0 & 1.0 & 0.70 \\ 0.35 & 1.0 & 1.0 & 1.0 & 0.75 \\ 0.4 & 1.0 & 1.0 & 1.0 & 0.75 \end{bmatrix}, "REQ_PLAN.XLS", \quad (5)$$

while works w_3 , w_4 are rejected as unauthorised with the results $\tilde{Y}_3 = \{\mathbf{0}, "DUMMY"\}$, $\tilde{Y}_4 = \{\mathbf{0}, "DUMMY"\}$. System state matrix $\Omega_a(t_1 + \Delta t)$ and rejected works matrix $W_{\Pi_a}^z(t_1 + \Delta t)$ apparently look like shown on Fig. 8.

At the next time point t_2 **PM** agent receives the task

$$W_{PM} = \{ w_1 = ('Assemble project proposal', X_1, Y_1), \\ w_2 = ('Choose best DB Schemata Plan Bid', X_2, Y_2), \\ w_3 = ('Choose best Software Model Plan Bid', X_3, Y_3), \\ w_4 = ('Choose best REQ Analyses Plan Bid', X_4, Y_4) \}$$

Θ_{DBA}	0	0	0	0	0	-1	0	0	0
Θ_{PROG}	0	0	0	0	0	0	-1	0	0
Θ_{PM}	1	0	0	0	0	0	0	0	...
Θ_{DOC}	0	0	0	0	0	0	0	-1	0
Θ_{TST}	0	0	0	0	0	0	0	0	0
Θ_{LIA}	0	0	0	0	-1	0	0	0	-1

with

$$X_2 = \{ \tilde{Y}_5^{DBA}, \tilde{Y}_5^{LIA} \}, \\ Y_2 = \{ y_2^1 = (\text{Possibility}(\text{budget}, \text{duration})), \\ y_2^2 = (\text{Plan_File_Name}(\text{Plan_Template})) \}; \\ X_3 = \{ \tilde{Y}_6^{DBA}, \tilde{Y}_6^{PROG}, \tilde{Y}_6^{LIA} \},$$

$$Y_3 = \{ y_3^1 = (\text{Possibility}(\text{budget}, \text{duration})),$$

$$y_3^2 = (\text{Plan_File_Name}(\text{Plan_Template})) \};$$

$$X_4 = \{ \tilde{Y}_7^{DBA}, \tilde{Y}_7^{PROG}, \tilde{Y}_7^{LIA} \},$$

$$Y_4 = \{ y_4^1 = (\text{Possibility}(\text{budget}, \text{duration})),$$

$$y_4^2 = (\text{Plan_File_Name}(\text{Plan_Template})) \};$$

and is ready to perform the works w_2 , w_3 , w_4 . Work w_1 is again delayed and scheduled for the next modelling step. The task of **PM**'s macromodels for w_2 , w_3 , w_4 is actually

Fig. 9. System state (related to W_a) at $t_2 + \Delta t$.

Θ_{DBA}	0	0	0	0	0	-2	0	0	0
Θ_{PROG}	0	0	0	0	0	0	-2	0	0
Θ_{PM}	0	-1	-1	-1	0	0	0	0	...
Θ_{DOC}	0	0	0	0	0	0	0	-2	0
Θ_{TST}	0	0	0	0	0	0	0	0	0
Θ_{LIA}	0	0	0	0	-2	0	0	0	-2

Fig. 10. System state (related to W_a) at $t_3 + \Delta t$.

to choose optimal plans from the accepted bids according to the provided *Possibility* feedbacks. In our case the task is rather simple as far as only one positive bid per each work has been received: **LIA**'s for w_2 , **DBA**'s for w_3 , and **PROG**'s for w_4 . The results of this step are that presented on Fig. 9. Note that functional components **DBA**, **PROG**, **LIA** are idle within discussed interval $[t_2, t_2 + \Delta t]$, though they are capable to perform some works.

The similar optimisation work w_1 is performed by **PM**'s macromodel within $[t_3, t_3 + \Delta t]$. The work is to assemble the resulting project plan using plans provided by other community members ($\tilde{Y}_2, \tilde{Y}_3, \tilde{Y}_4, \tilde{Y}_8, \dots, \tilde{Y}_{15}$). System state is presented on Fig. 10. The resulting *Possibility* (pessimistic estimation) is as follows:

$$\tilde{y}_1^1 = \min(\tilde{y}_2^1, \tilde{y}_3^1, \tilde{y}_4^1, \tilde{y}_8^1, \dots, \tilde{y}_{15}^1) = \begin{bmatrix} 0.1 & 0.15 & 0.4 & 0.6 & 0.3 \\ 0.1 & 0.5 & 0.4 & 0.75 & 0.35 \\ 0.2 & 0.75 & \textcircled{1.0} & 0.8 & 0.35 \\ 0.3 & 0.9 & 1.0 & 0.8 & 0.35 \\ 0.35 & \textcircled{1.0} & 1.0 & 0.85 & 0.35 \end{bmatrix} \quad (6)$$

Though the task W_a at t_4 has already been accomplished one more modelling step is needed for to check if the agents have stopped to absorb incoming works. Final step results are shown on Fig. 11.

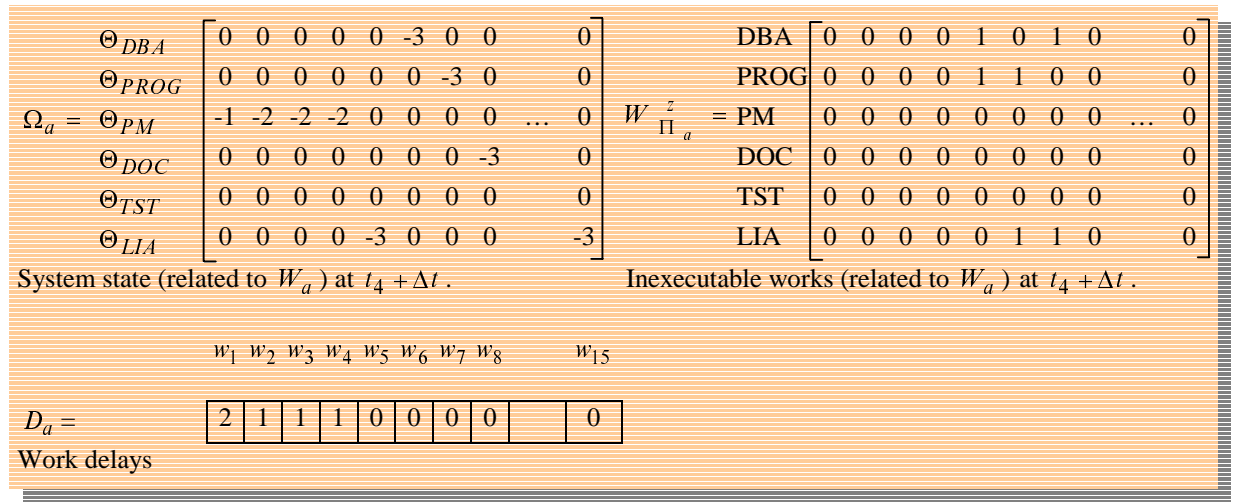


Fig. 11. System state, inexecutable works and work delays (related to W_a) after process W_a has been accomplished.

3. Case Study Analysis

If we analyse the results of Planning Process modelling we may figure out the following:

1. Parametric feedback for modelling of co-ordination and negotiation. The resulting *Possibility* (6) shows that the proposed project may be unconditionally performed with increased budget (1.8b) even in shorter time then it is requested - 0.8d. In case project budget is not subject to be increased the project may be accomplished in requested time with budget starting from 1.0b. To say generally, parametrical feedbacks provided by the framework make modelling of co-ordination (like CNP or brokering) or negotiation (like auction) protocols by means of work sequences quite simple.

2. Learning and behaviour evolution. $W_{\Pi_a}^z$ analysis shows that works w_5 , w_6 , w_7 had been rejected by some agents, though they have finally been performed. The inferences are: $W_{\Pi_a}^z$ may be further used as a tip for more accurate work assignments in the modelling of planning processes - agent **PM** learns about the capabilities of other community members related to the process discussed and thus adapts its behaviour. We may consider work w_i to be inexecutable by the agents community involved into the process Π_a only if the column i of $W_{\Pi_a}^z$ doesn't contain zero values. There is however no answer to the question about the behaviour of the community in case new member agents with new capabilities (for instance partially covering works already placed to $W_{\Pi_a}^z$) are entering the team. For the moment it is assumed that since the process has been launched to execution and until it is accomplished the contingent of functional components is fixed.

3. Agents' load evaluation. Evaluation of agents' activity related to process Π_a - refer to diagram on Fig. 12. - shows that average agents' load is 37.5%.

4. Planning by simulation. One of the accessory results which may be automatically obtained from Planning Process simulation is the draft of proposal preparation plan (see Fig. 13.).

5. Communication acts as work sequences. Communication acts among the agents performing the process are modelled as sequences of certain works performed at different time points. In our case the execution of work w_3 is formally classified (see Section 1.) as multicasting undetermined request w_6 to **DBA**, **PROG** and **LIA** with further results analyses by **PM** (Fig. 14a). Our case study, however, shows that it is reasonable to perform the works w_3 and w_6 in reversed order to avoid the necessity to synchronise the influence with corresponding reaction(s). As

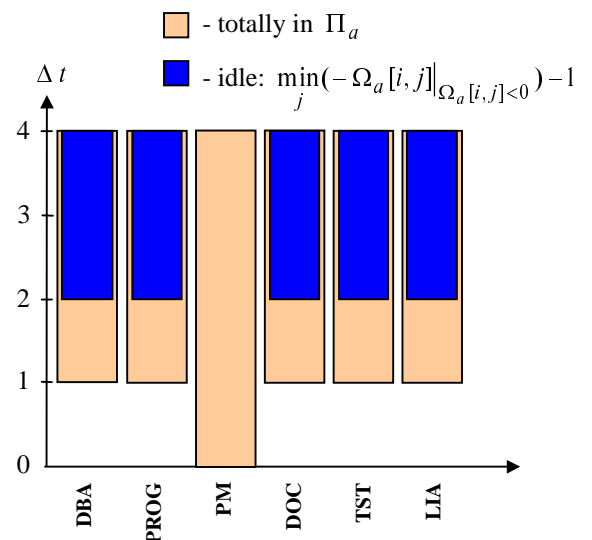


Fig. 12. Agents activity within Π_a .

Task/Work	t_1	t_2	t_3	t_4
Tasks Specification	PM			
REQ Analyses Planning		LIA		
DB Schemata Design Planning		DBA		
Software Model Design Planning		PROG		
Programming Planning		PROG		
.....		...		
Customers' Training Planning		LIA		
REQ Analyses Plan Optimisation			PM	
Software Modelling Plan Optimisation			PM	
DB Schemata Design Plan Optimisation			PM	
Project Plan Assembly				PM

Fig. 13. Automatically generated plan of modelled Proposal Preparation process.

it is shown on Fig. 14b these works are performed, say informally, in "inverse polish" order. Another thing to be mentioned is the mechanism of setting up appropriate delays for the elements of the work 'stack'. In our case the framework uses incoming parameters verification routine F^X for delaying the execution of work w_3 . It will not be performed until valid values of $\tilde{Y}_6^{DBA}, \tilde{Y}_6^{PROG}, \tilde{Y}_6^{LIA}$ are provided - i.e. until **DBA**, **PROG** and **LIA** complete the execution of work w_6 multicasted by **PM** within the time interval $t_1 + \Delta t$ (see Fig. 14b). In case the works do not have parameter relationship the framework provides alternative mechanism for ordering the sequence - agent's state constraints (see Fig. 14.). The process of changing the state is the application of some constraints having definite lifetime and looking as follows:

$$s_6 : \left\{ \begin{array}{l} \dots \\ \text{AddConstraint} = \text{'Reject } w_3 \text{'}, \text{LifeTime} = 1 \\ \dots \end{array} \right. \quad (7)$$

in case multicasting of w_6 was a communicative act of *directive* type.

6. FIPA compliance. The work sequence model presented on Fig. 14. is notably similar to FIPA Contract Net Protocol [7]. The conjectures are that 1-st - we may possibly model FIPA ACL communicative acts by means of presented framework and 2-nd - we may possibly use ACL as the transport language for inter - agent communication within the communities modelled in frame of the proposed approach.

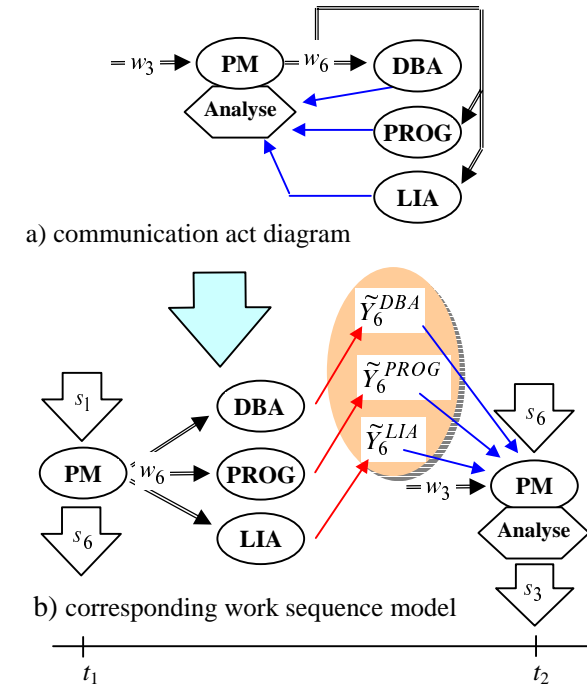


Fig. 14. Modelling of *undetermined request with results analysis*.

modelling the process of IS development proposal preparation planning has shown that the approach may be applied to modelling business processes as tasks presented by sets of atomic works. Case study analysis has provided some results valuable both from application and theoretical generalisation points of view. The operational model provides good opportunities for further fine-tuning of community members behaviour (ontologies, state constraints and macromodel programs) based on the utilisation of the results obtained while modelling of one or another task execution process. The framework as well provides visible possibilities to model known communication protocols and communicative acts (contract net, negotiation, ...) by means of appropriately configured work sequences.

The plans for further research and development are as follows:

7. Manual preparation work. Even this simple case shows that in practice we need to apply lots of inhouse work before the functional system is ready to operation. Macromodels, ontologies, state constraints, parameters', results' and role descriptions are to be prepared manually and stored into the agent/community knowledge base. The positive thing is anyway that the major part of this work is performed once and the result may be re-used by another agents/communities (performing different tasks). Another thing to be mentioned is that not all of the atomic works may be executed automatically by the macromodel program. The agents are used as personal assistants providing substantial aid to the execution of the routine part of work. Execution of work w_6 , for instance, assumes the participation of a human 'master' with his creative mind for at least the informal part of the plan preparation work.

4. Conclusions and Future Work

The theoretical framework [4] for modelling the processes of information interchange among the members of task-oriented evolving agent communities has been applied to modelling of one of the practically important types of business processes - a Planning Process. The case study of

More case studies. The case study presented is just one practically important case of the framework approbation. More cases for various types of business processes (supply chain, auctions) are planned to 1-st - verify framework applicability and to 2-nd to possibly enhance the models if the necessity arises.

Architectural framework. The layered architectural framework utilising the theoretical models of [4] is under development and will soon be deployed. The layers of the agents community architecture will comprise: generic agent architecture, agent-to-agent interface specification, communicative acts specification, task-oriented agents community specification comprising agents specialisation and feedback co-ordination method, ontology representation, knowledge sharing and interoperability issues.

Implementation. The final goal is to implement Agent-Based IS for Managing Business Processes for a University/Enterprise Department. One of the strong candidates we are considering now is Zaporozhye State University Publishing Department.

References

1. N. R. Jennings, P. Faratin, M. J. Johnson, P. O'Brien, M. E. Wiegand: "Using Intelligent Agents to Manage Business Processes", Proc. First Int. Conf. on The Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM96), pp. 345-360. London, UK.
2. Ouksel, A. M.: "A Framework for a Scalable Agent Architecture of Cooperating Heterogeneous Knowledge Sources" in Intelligent Information Agents: Agent-Based Information Discovery and Management on the Internet, ed. M. Klusch, Springer, 1999, pp. 100-122.
3. K. Sycara: "In-Context Information Management through Adaptive Collaboration of Intelligent Agents" in Intelligent Information Agents: Agent-Based Information Discovery and Management on the Internet, ed. M. Klusch, Springer, 1999, pp. 78-99.
4. S. U. Borue, V. A. Ermolayev, V. A. Tolok: Diakoptical Approach to Modelling of Processes In Multi-Functional Information Systems.// "Artificial Intelligence" - a theoretical journal №2, 1999, ISSN 1561-5359, spec. issue: Proceedings of International Conference "Knowledge-Dialog-Solution" - KDS'99. Katsiveli, 13-18.1999, pp.211-219 (in russian)
5. Kron, G., Diacoptics. Macdonald, London, 1963
6. Бутырин, П. А., Борю, С. Ю. Комплекс программ макро моделирования систем: Сб. Моделирование силовых вентильных преобразователей., Киев ИОД, 1989, с. 12-20
7. Foundation for Intelligent Physical Agents (FIPA) Spec: DRAFT, Version 0.2, Agent Communication Language, 1999, <http://www.fipa.org>
8. Finin, T. and Fritszon, R. KQML - A language for protocol and information exchange, Proc 13th DAI workshop, pp 127-136, Seattle, WA, USA.
9. Rao, A. S. and Georgeff, M. P. BDI agents - from theory to practice, Proc. of the 1st Intl. Conf. on Multi-agent Systems, San Francisco, 1995.
10. Jennings, N. R., Sycara, K., Wooldridge, M., A Roadmap for Agent Research and Development, Autonomous and Multi-Agent Systems, 1, 1998, pp. 7-38.