Default Interactions for Multi-Agent Simulations of Complex Organizations

Nick Szirbik[1], Marco Stuit[1]

[1] Department of Business and ICT, Faculty of Economics and Business, University of Groningen, Landleven 5, 9747 AD Groningen, The Netherlands
{N.B.Szirbik, M.Stuit}@rug.nl

Abstract. Default interactions are a novel concept that supports the agent-based simulation framework and modeling toolset that is built around the TALL language. This paper argues that default interactions are ubiquitous in business processes that are mostly constituted and driven by human interactions. Therefore, it is necessary to make the default interactions explicit by developing and imposing protocols for their execution. The paper shows how interactions that otherwise need a very complex execution scheme can be radically simplified by employing various types of default interactions, depending on the type of the organization that is modeled and simulated.

Keywords: Organizational interactions, Business processes, Agent-based simulation, Interaction protocols

1 Introduction

Agent-based simulation of organizations can be used for multiple purposes, among others, organizational (re)design, a better understanding for its members in the case of participative simulation, and also for multi-agent system (MAS) iterative development [1]. In order to perform simulations, a formal model of the organization and its members should be enacted. To develop a "complete" model is impossible, because the real world contains unpredictability and non-computable functions. However, a complete model is not required since human intervention, in the form of interactive simulation, can fill the gaps in the models.

In this paper, organizations and their processes are viewed as *interactive systems*. An interactive-centric perspective [2] simplifies the study, specification, formal modeling, and simulation of complex organizations. Interaction is taking place between concurrent entities, and a vast body of knowledge, languages, and tools emerged over decades of research in computational concurrency. Organizational simulation and MAS design applied formalisms such as CSP [3], CCS [4], Pi-calculus [5], process algebras [6], [7], the actor model [8], and conversation models [9]. However, all these approaches lack something that Wegner [10] calls *constructive models of interaction*. He argues that a distinctive paradigm shift is necessary within (or out of) concurrency formalisms, a shift that makes the explicit modeling of interaction a striking hallmark.

The research presented here builds on previous work on organizational modeling and simulation, based on the concepts of *interaction composition* (enacted via Interaction Structure diagrams) and abstract behavior descriptions (via Agent Behavior diagrams), both being formalized in a language named TALL (The Agent Lab Language, see [11]). The main contribution of this paper is an extension of the TALL-based modeling and simulation with the concept of *default interactions*. This is a concept inspired from organizational theory, and also has a formal basis in communication protocols and agent interaction protocols [12]. Default interactions provide an execution anchor to the structure of interactions that are performed as part of the agent-based organizational simulation. The main point this paper makes is that the explicit visualization of default interactions in the modeling and simulation methodology simplifies the work of the organizational modeler especially in terms of process descriptions.

The paper is organized as follows: the next section provides background information on interaction-centric agent design methods in general and the need for TALL. Section 3 introduces an approach based on interaction plans and explains how an organizational process can be set up for execution (in reality and simulated) by using composite interactions. Patterns for default interactions are identified (Section 4) and interaction protocols for these are elaborated (Section 5). Section 6 discusses the methodological implications of the default interaction, and Section 7 summarizes our contribution.

2 Background

Interaction is a central issue in designing large organizations, or their simulation models, that execute concurrent processes in a dynamic and collaborative environment. Such organizations support communities of people that perform complex activities and have goals that are difficult to understand and model. Making the interactions between the people explicit enables the design of interaction "scripts" and allows their composition for process execution.

Any inter- or intra-organizational process that can be reduced to a set of interactions tends to be regulated, at least partially, by documented patterns that emerged during repeated execution of these interactions. These artifacts or interaction "scripts", which are very similar to communication or interaction protocols, are built and sometimes enforced to ensure a disciplined way to improve the execution of business processes over time. There is both a conceptual distinction and overlap between communication protocols and interaction protocols. The former insist on the data formats in the information exchange process, plus the order in which the information is sent and received. The latter insist also – adding to the communication aspects – on the specific activities and their relation to each other that have to be performed by the various participants in the interaction.

In the agent research and development community, the importance of the interaction protocol has been recognized early [13] and many agent frameworks (e.g. Gaia [14] and MESSAGE [15], [16]) include recommendations for notations depicting protocols, or even allow for atomic representations of protocols. Some

agent-oriented modeling languages have diagram types for explicit protocol description, like AUML [13] and AORml [17]. Protocols are typically modeled in UML-like diagrams, like sequence diagrams (which do not insist on the internal activities but more on the message exchange), activity diagrams (which show the activities the agents should perform, allowing for internal parallelism), and even statecharts [12].

In [12] a three-layered approach to agent interaction protocols (API) in the AUML language is proposed. On the first layer (called packages or templates, using concepts and constructs from UML), the structure of the interaction is described, and on the second (interaction diagrams), the behavior of the participating agents is enforced. The third layer is the internal agent processing description. Although the AUML approach follows model-driven development, it is not suited for building effective simulation models for organizations because of the lack of a formal foundation. Previous work on TALL introduces graph theory and tree combining algorithms for the first layer ([11]), and Petri net extensions for the second layer ([18]).

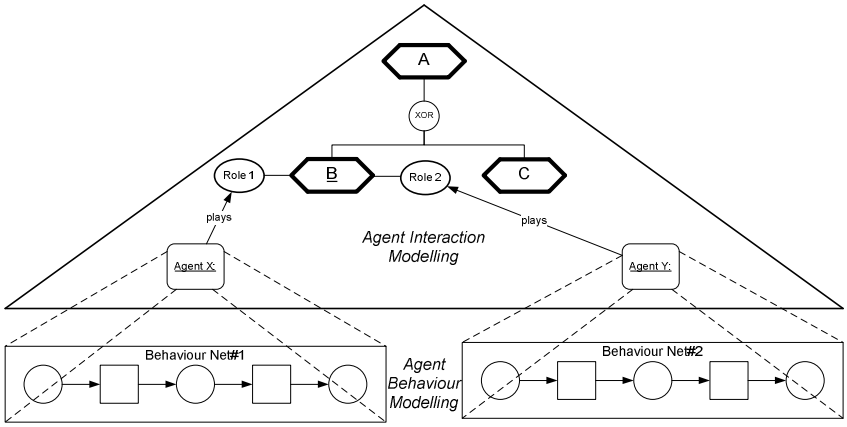

Fig. 1. The basic constructs of the TALL modeling language, depicting interaction and behavior modeling.

TALL is used for organizational modeling, business process simulation, and MAS development. In the language, agents have beliefs about how the interactions should be executed. Two types of diagrams capture two types of agent's beliefs. First, Interaction Structure (IS) diagrams show a tree structure of interactions including composition and dependency relationships between the interactions. The IS diagram represents a partial or complete local interaction-centric process belief. Agents can have different interaction-centric beliefs. This results in multiple local IS diagrams. A set of local IS diagrams can be merged into a mutually agreed global IS diagram using the approach described in [19]. An IS diagram does not prescribe how each interaction is to be executed by the agents, only the way they are structured and ordered for interaction execution. Second, the Agent Behavior (AB) diagram captures behavioral-centric process beliefs. This diagram is in fact a sort of a protocol, but one seen from the perspective of an individual agent. Formally, in TALL, an AB diagram

is a swimlaned Petri Net [18] in which each swimlane corresponds to a role, and contains the activities that are to be performed when playing that role. The swimlanes are connected through message places, which denote the interaction's exchange points. When taking part in an interaction, an agent intends to execute the swimlane that it is marked with the role name the agent is playing in the interaction, and expects the other agents to do what is described on the other swimlanes in accordance with the roles these agents are playing. In essence, the execution of an interaction in the IS diagram is the coordinated execution of all the AB diagrams owned by the participating agents.

In Figure 1, interaction *A* can be de-composed in two other sub-interactions, and a routing construct (XOR – mutual exclusion, SEQ – strict sequence, PAR – parallel execution) specifies the order of their execution. Each interaction has placeholders for role names, and agents can play these roles. In order to execute interaction *B* in Figure 1, agent *X* owns an AB diagram (illustrated by the Behavior Net#1) and agent *Y* owns another AB diagram. The message exchange points are not shown. The coordination or composition of the two behaviors for interaction *B* should yield a coherent process description (e.g. a Petri net that is sound).

In this paper, the agent concepts of *knowing* and *believing* are used loosely. In general, "knowing" refers to globally shared knowledge, and "believing" refers to local knowledge. For the sake of clarity, it is emphasized that in this paper the verb *knows* refers here to the agents' local perspectives as well.

3 Interaction Plans

It is possible for an interaction to have a pre-defined protocol. A pre-defined protocol includes a pre-defined process description for each role that is involved in the interaction. In other words, it acts like a central process definition that regulates the behaviors of the participating agents. If all the interactions that are known in a certain agent domain (i.e. an organization with clear boundaries) have such a centralistic protocol, the way to plan an entire process instance is just to enact a global IS diagram that was agreed upon by all the participating agents. Completion is a bottom-up process in the IS diagram. Sub- or child interactions complete their parent interaction taking into account the specified routing. This means that the entire process can be executed when centralistic protocols are available for all the leaf interactions in the IS diagram.

However, when centralistic protocols are used to perform the interactions, the individual beliefs of the agents (i.e. their AB diagrams) are actually not used since their behavior is regulated externally. This is not characteristic of an agent-oriented approach. On the other hand, the centralistic protocol can be overruled and the agents can still use their AB diagrams.

The research question this paper addresses is: "How to enact an interaction plan when there are no centralistic protocols, or when only a few of the interactions have some?" The agent paradigm states that agents should have a certain degree of autonomy. TALL is based on the premise that even for interactions with centralistic protocols, the participating agents have the power to override the protocol in certain

circumstances. That means that the agents have to execute somehow the interaction, based on their own (previously learned and tested) behaviors. The problem is that these agents can have very different beliefs about how the others will behave.

3.1 The pre-planning of an interaction vs. on-the-fly execution

Suppose that an agent, called Initiator Agent (InAg), knows in a certain context (or it is externally triggered) that it has to execute a single interaction, which is to become part of the execution of an overall process instance that is defined in a global IS diagram. Assume that there are no centralistic protocols pre-defined for this interaction. What the *InAg* can do is to use a belief (in the form of an AB diagram) that has been used previously by him for this interaction (at least if *InAg* has such a belief or experience). There are two possible scenarios to proceed from here.

The first scenario is that the agent initiates this behavior and just sends the first exchange message defined in the AB diagram, not to a specific agent, but to a role. In this case, the environment (where all agents coexist and are recognized) makes sure that a target agent is reached, by attaching an agent to the role. In this situation, the InAg expects that the other agents will do their part, and the interaction will end somehow. In the case of deadlocks or unfinished interactions, the environment has the responsibility for ensuring the overall coherence of the process execution. Typical for this scenario is that agents are assigned to the roles on the fly, as messages are sent to the roles and agents to play the roles have to be found dynamically. The environment's mechanism that insures this kind of scenario offers more flexibility, but it is more difficult to implement.

The second scenario depends on a mechanism that is less flexible, but it is much more robust and easy to implement. First, the agents that will play the roles are looked for and assigned to the roles. After, a plan for the execution of the interaction is build collaboratively by the selected agents. In this paper, the focus is on the second scenario.

What can go wrong in the first scenario? For example, *InAg* can own an AB diagram that does not start with sending a message, but receiving one. Then, *InAg* has to pass the role of Initiator Agent to an agent that is available and authorized for this interaction and also owns an AB diagram that initiates the first exchange in this interaction. To find this agent, an over-complicated procedure has to be executed. Even if *InAg* is the first to send a message, it is uncertain that the other agents that are triggered to become part of this interaction will have AB diagrams that are aligned in a way that ensures a coherent execution of the interaction. In [18], there is an example of how alignment of behaviors can be achieved on the fly, but the research in this area is still in its infancy.

The robustness of the second scenario is insured by two facts: even before the start of an interaction, *InAg* (and not the environment or a hierarchically superior agent) makes sure that all the agents that need to participate are found and committed, and moreover, a plan for interaction execution is agreed upon before the interaction starts. In this way, each interaction that is to be executed for the overall process instance execution will have either a pre-defined plan defined in the form of a centralistic protocol or a plan that is agreed upon by the agents just before a specific interaction

starts. However, in both cases, the agents are assumed to be aligned and in agreement, and will know how to execute the interaction. This is a strong assumption and in reality, agents will have slight or serious misalignments between their ABs.

3.2 Using the Escape/Intervention Mechanism to achieve always alignment

In order to solve misalignments easily, the agent paradigm offers a simple mechanism, that is, the *Escape/Intervention*. In previous research [1], it is shown how this mechanism is formally described. Succinctly, *Escape* is a mode of an agent which is triggered when the agent is not able to proceed with its actions. This mode is triggered for example by trying to push over explicit constraints (like "if a sales order is bigger than $1000, you have to report it to the senior manager). In other scenarios, the trigger is due to time escalation ("if I do not know what to do and nothing happens in the next half hour, I am alerting my boss"). Triggers can be established on more complex mechanisms (like the fuzzy concept of "the level of trust I have in the other agent I am interacting with now is too low, thus I am asking my boss how to proceed"). The latter ones are notoriously difficult to formalize and implement in agent-based software systems.

In the context of the enactment of interaction plans, the *Escape* mode can be invoked when:

- *InAg* cannot find the agents to execute the interaction;
- Even after finding all the agents, nobody knows how to execute the interaction, that is, there are no AB diagrams for this interaction in the belief bases of these agents;
- the gathered agents cannot agree on the plan (without asking for help).

These three situations are typical for many business processes. The *Escape* mode ensures that in most of the foreseeable situations that block the execution of the process instance, a higher authority (called the Deus ex Machina in [1]) intervenes and solves the problem.

If there is no trigger for *Escape*, the other part of the mechanism, namely Intervention, enforces that the execution is to a certain degree monitored by the Deus ex Machina (which can be just another agent, but with higher authority) that has the power to intervene when deemed necessary. The purpose of the intervening agents is to help the "escaping" agents to build a plan. For example, in an interactive (i.e. supervised, or "piloted" by human players) agent simulation where two agents cannot align their ABs, and it is automatically detected that there is no alignment resolution, the human players can take over and design ad-hoc a plan for the interaction that ensures alignment. This plan is stored by the simulation agents in their belief bases, and can be re-used in future interactions automatically. In a real organization, when two lower level human actors cannot agree how to proceed, they can ask their superiors to help them and sort out a procedure, or the superiors, via reporting and monitoring, perceive that there is a problem and intervene.

From an agent perspective, it is desirable to have a low rate of *Escape* and *Intervention*, because an organization or a system that needs high-level guidance too

often and/or needs repeated *Intervention* cannot be considered really autonomous. In the remainder of this paper, it is shown how the introduction of default interactions leads to a better rationale and implementation of the Escape/Intervention mechanism.

3.3 Identifying the most basic default interactions

Default interactions benefit and improve interaction set-up and execution by standardizing and making explicit the procedure to build interaction plans. Any interaction in a TALL IS diagram, when to be executed, will follow such a procedure. In this paper, the procedure is also visualized as an IS diagram. The interactions in this IS diagram can have easily defined protocols, and are the logical candidates for default interactions.

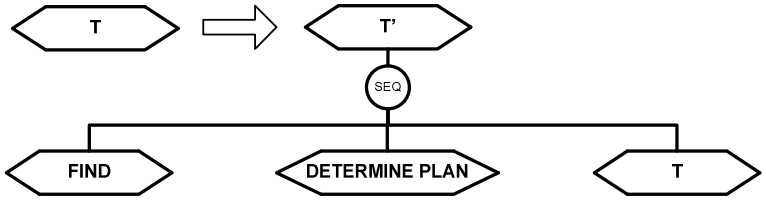

Fig. 2. Interaction *T* is transformed into *T'* that contains two default interactions.

In Figure 2, the procedure for interaction *T* is made explicit in the IS diagram *T'*. The diagram *T'* contains three sub interactions that have to be solved in a sequence, *FIND*, *DETERMINE PLAN*, and the original *T*. These candidates for default interactions will have a special role *Initiator*. The agent that initiates *T* will always play the *Initiator* role. This role appears on all the default interactions.

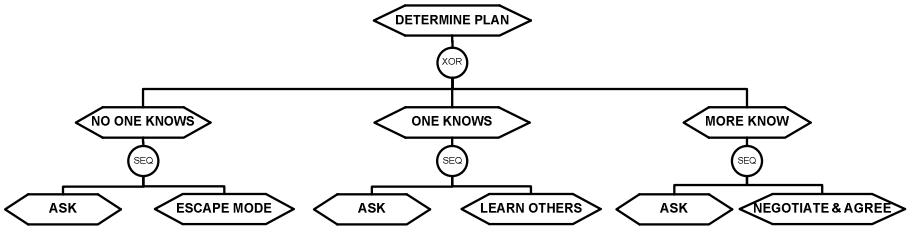

Fig. 3. An overview of the *DETERMINE PLAN* interaction tree.

During *FIND* the initiator is identifying and selecting the agents that will play the roles in *T*. After the agents commit to play these roles, *DETERMINE PLAN* ends with an agreed upon plan. Based on this plan, interaction *T* has a higher chance to end successfully. Figure 3 presents an overview of the three alternatives that can emerge during the execution of the *DETERMINE PLAN* default interaction. The alternatives are mutually exclusive and each has a default interaction named *ASK* as its direct child that is to be executed first. This is due to the fact that an IS diagram is

performed bottom-up in terms of composition, and performed left-right in terms of dependency.

4 Refining the Default Interactions

4.1 Case 1: No One Knows

The first alternative for *DETERMINE PLAN* is when no agent has been found that has a viable behavior or AB diagram that can be used as a plan for interaction execution. In order to reach this conclusion, the first sub interaction of any of the three alternatives for *DETERMINE PLAN* is always *ASK PARTICIPANTS*. A protocol for this interaction is proposed in Section 5. In this branch of the tree (see Figure 3), there are no agents who know (including *InAg*). There are two obvious solutions. The first solution is to go into *Escape* mode (either the whole group of agents linked currently to the interaction, either the initiator only), or to backtrack in the tree of Figure 2 to the *FIND* interaction and start everything from scratch. A more complex solution is to try to find separately an agent that owns the appropriate AB diagram, without dismissing the first group, but this is actually very close to going into *Escape* mode. This first alternative *for DETERMINE PLAN* can be depicted as an IS diagram with *NO ONE KNOWS* as root, like in Figure 3.

4.2 Case 2: One Knows and Shares

The second alternative is when a single agent is found during the *FIND* interaction that owns an AB diagram, which covers all the roles in the interaction. In other words, this agent not only has an explicit description of its own intended behavior but also has explicit expectations for the behaviors of the other involved roles. In a sense, this is the best case scenario.

This alternative is represented as the middle branch in Figure 3 and has two default sub interactions. The first sub-interaction is the same *ASK* default interaction mentioned in the first alternative. The second sub-interaction is another default interaction that ensures that the 'knowledgeable' agent shares its explicit expectations with the agents that were selected for the other roles, and ensures that the agents agree to execute the interaction according to the unique AB diagram. This AB diagram becomes their common plan – it can be also interpreted as a one time centralistic protocol. The implementation (in an agent simulator for example) of this alternative is the simplest out of all alternatives. When this alternative is viewed in the context of organizational theory, one can say that the agent who knows is teaching the other agents what to do. If the agents keep the behavior that was learned (i.e. these agents store their own AB diagram based on this behavior), this can be regarded as organizational learning and knowledge diffusion.

4.3 Case 3: Two or More Know

The third alternative yields the default interactions depicted on the right-hand branch in Figure 3. In this alternative, the *ASK* default sub-interaction finds that more than one agent owns an AB diagram for this interaction. Here, the best case scenario is when all the AB diagrams completely overlap (i.e. full alignment), and they can proceed immediately to execute *T*. However, some agents can have similar AB diagrams and others can have no AB diagram at all. This leads to the same ending as the second alternative – where the ones who know, teach the ones who do not know. A particular scenario is when everybody in the group owns an AB diagram that contains only the swimlane with the intended behavior (activities and message exchange points) of the owner. In this scenario, everybody knows what to do in their particular role, but nobody knows what the others are doing. Before putting together these swimlanes into a single plan it is necessary to make sure that the message exchange points are aligned and the plan is coherent.

The most complex scenario, which is difficult to formalize by using default interactions, is when the agents in the group own different AB diagrams (i.e. everybody has a different belief about how to execute the interaction). This complex situation appears in many organizational settings, and it is rooted in human psychology and simple economics. Human agents tend to have mental models of their behavior in which they move activities that they do not like in the swimlanes of the other roles. This creates disagreements like "You should do this, not me", which are difficult to manage, especially when the level of trust and understanding of the others is low and the organization does not sanction properly manipulative behavior of its members. Future research will investigate how default interactions can integrate different AB diagrams into a fair and coherent plan, and how agents can finally agree on it.

5 Protocols for The Three Basic Default Interactions

Default interactions can be used during interactive simulations of agents or in a real organization only if the agents executing the interactions are all aware of the set of default interactions. In other words, agents that are involved in a default interaction need to follow the description of this default interaction as a protocol. Due to their simplicity, the method to describe the default interactions should use the same formalism that is used for AB diagrams. In this way, default interactions become full-fledged interaction protocols – obeyed by all the agents. In the next subsections, a few examples of protocols for default interactions are presented.

5.1 The protocols for the *FIND* default interaction

It is obvious that one of the most important default interactions is *FIND*. Any organization that is viewed (or simulated) as an agent system needs a protocol that ensures that for each interaction, *InAg* is able to discover the identities of the agents that are capable and authorized to play the involved roles.

FIND can be seen as a composite interaction, as depicted in Figure 4. The default protocols that should be enforced for *FIND* are presented in Figures 5 and 6. Note that this is only a preliminary investigation of the *FIND* interaction, because other protocols to find agents exist. For example, it is possible that an interaction that is to be started by *InAg* already has all the other agents linked to the roles they have to play based on historical information. The latter scenario ends the *FIND* interaction immediately.

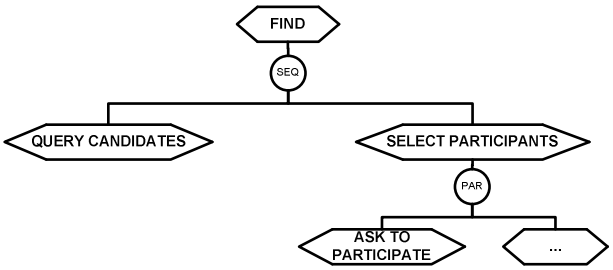

Fig. 4. Overview of the *FIND* interaction tree (The interaction with dots represents the other potential interactions that can take place during the selection of agents).

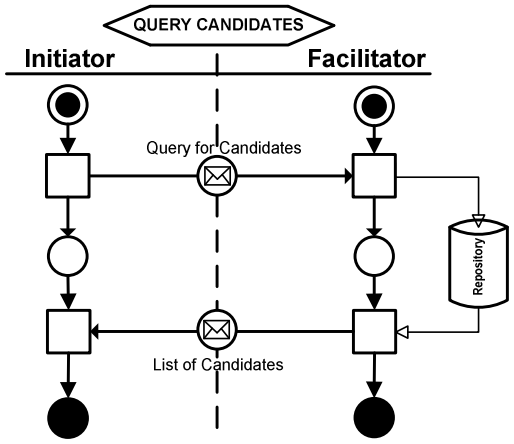

Fig. 5. Protocol of *QUERY CANDIDATES*.

5.2 Protocol for the *ASK* default interaction

The *InAg* agent has to make sure that the information necessary to construct an interaction plan exists when there is no centralistic protocol available. The interaction *ASK PARTICIPANT WHO KNOWS* appears as a default interaction on all the three

branches of the *DETERMINE PLAN* default interaction. The default protocol for this interaction is presented in Figure 7.

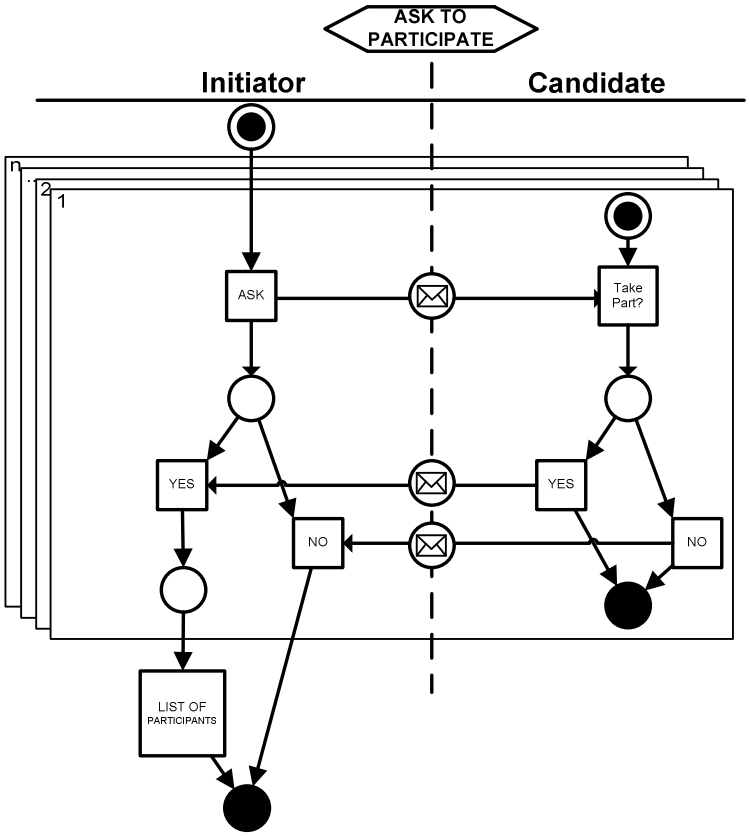

Fig. 6. Protocols of *ASK TO PARTICIPATE*.

5.3 Informal description of the protocols for the default interactions

The default interaction ASK WHO KNOWS can be transformed easily into a default interaction that leads to agreement. This interaction, called AGREE, has the following sub alternatives – for which the protocols are not developed (and depicted in this paper):

1. everybody knows how to (owns an AB diagram) and all AB diagrams are aligned. Interaction *T* can start immediately;
2. everybody knows its own role-swimlane and the result of putting them together is a coherent plan. Interaction *T* can be started;

3. some know and all the existent AB diagrams are aligned. The default interaction LEARN OTHERS has to be triggered and finished before *T* is started.
4. some know but the ABs are not aligned. In this case, an alignment mechanism or the Escape/Intervention is necessary.

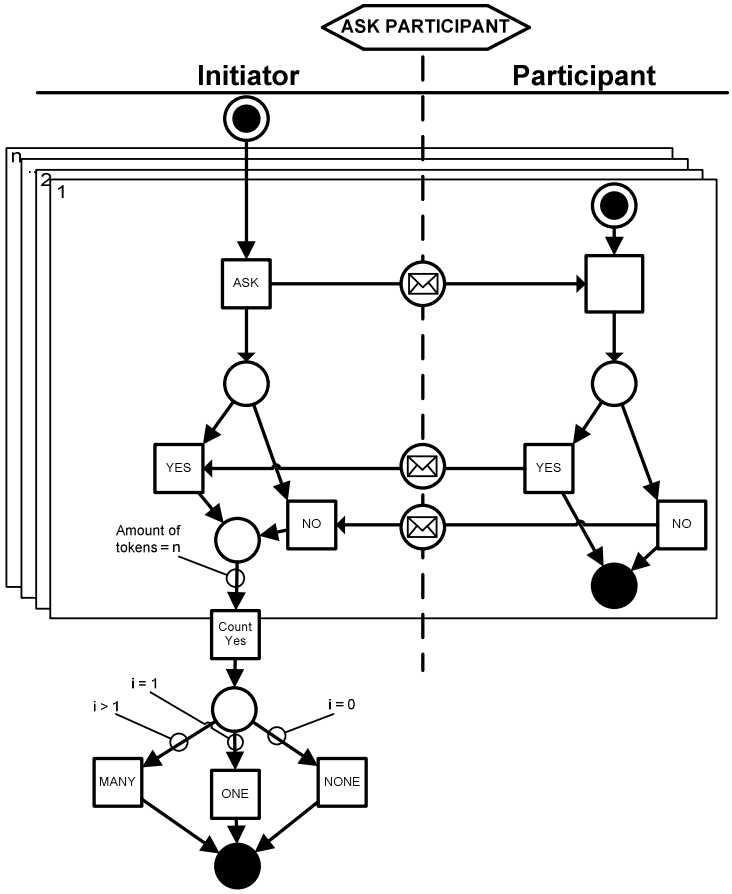

Fig. 7. ***ASK*** participants who knows how to interact.

In future research, it will be investigated how a default interaction/protocol will look like in the situation where everybody owns a different AB diagram. A simple method is to assign weights to various activities that appear in the AB diagrams, and re-distribute the activities among the participating agents in a way that gives approximately the same sum of weights on each swimlane (for fairness), but does not assign activities to agents that they are not able to perform. For example, it makes sense to move the activity "translate from German to English" from one role to the other, but only if the agent taking over this activity has the skill to perform such a translation. This requires a supplementary level of facts assigned to the agents,

describing the skills they have and the activities they can perform, bringing the TALL models to a third level of process description granularity (the first being the IS diagrams, the second being the AB diagrams). However, this third level is not directly related to the process descriptions. Indeed, it is related to the agent descriptions (agent structures and their attributes) that have to be part of the agent simulation model but not of the process model.

6 Discussion

An attempt to simulate a complex organizational process should not necessarily imply a complex modeling exercise even if it results in very complex algorithms. The challenge in building simulations of complex organizations involves coping with the complexity that arises when composing even a non-trivial number of (even simple) constituents into a coherent functioning whole. The complexity of the resulting system arises out of the multitudes of combinations of ways in which the functioning of its constituents can mutually affect and interfere with one another [10].

The introduction of the default interactions in the TALL framework frees the modeling of processes in an agent organization from the necessity to represent routine or default interactions, which are complex, but are always the same. Default interactions do not need to be explicitly represented, and this leaves the expressiveness of the models uncluttered with interactions that appear almost within any agent interaction. Therefore, the new concept simplifies the modeling and simulation methodology. For example, the *FIND* default interaction is necessary anyway, every time when an interaction is triggered. From an organizational perspective, the environment should be designed in a way that supports this ubiquitous interaction. In a business organization, the list of personnel (containing organizational roles) is the first mechanism that an agent can use to find those who can be useful in a specific collaborative activity (i.e. an interaction).

If the designer of the simulation models selects a pre-planned manner of executing the interactions that compose a process, then the building of the simulation models is straightforward. A global IS diagram can be built that will perform the business process instance without the explicit representation of the default interactions: *FIND*, *ASK*, and *AGREE*. During the simulation, when the interactions in the global IS diagram are executed, the default interactions are automatically performed in the background for each interaction. Depending on the given context, the right alternative to determine an interaction plan is selected by executing default interactions like ASK. There are several caveats. Adorning an interaction-centric process execution will reduce the overall autonomy of the participants and lessen the “agentification” (or measure of delegation) in the organization. The execution becomes more predictable, but for sure less flexible or robust to environmental changes. The goal should be to keep a careful balance between the interactions that are executed solely using agent beliefs and the ones that are regulated using centralistic protocols. The simulation environment should provide a metric and a method to measure the rapport between “autonomous interactions” and “protocol-ruled interactions”. Another important point is that for different types of organizations, different default

interactions are predictably used. For example, in an adhocracy [20], where decision and control power is distributed among expert participants, the *FIND* default interaction and the *ONE KNOWS* alternative are heavily used. This allows the knowledge to diffuse, to the point that all the agents will finally have a view or belief about how the interactions in which they are party are performed. In bureaucracies, *FIND* is less important, because well established channels of communication ensure that the message exchange points are always aligned and everybody knows well, due to experience, exactly what he has to do (but it is not interested what the others are doing). In this case, the default interaction described in the second entry of the numbered list in Section 5.3 is the one that mostly applies. This is also due to the fact, that in a bureaucracy, horizontal interaction is limited. Most interactions will be vertical and they will be limited with regard to the number of participants (from employees to boss/manager).

The organizational structure that it is the most interesting for this kind of agent-based simulation is the flat, innovative organization. Here, all of the described default interactions will probably be used because in such organizations a lot of (horizontal) interaction occurs. This means that this organizational structure provides the best environment for refinement of the default interactions. Trying to solve interactions in this environment requires the introduction of more alternatives and more layers for the existing default interactions. Future research intends to focus mostly on these types of organization, with the goal to achieve realistic interactive simulations and also pave the way for the implementation of MASs that can support business processes within these organizations.

7 Conclusions

In order to perform agent-based simulations, or to insure agent-based support of a business process in an organization, a formal model of the process is necessary. As argued in this paper, this model is not required to be complete, in the sense that human intervention, in form of interactive simulation, can fill the gaps in the models. Although approaches have been developed in this respect, they lack a constructive model of interaction. The research in this paper discusses the necessity of default interactions to help set-up and agree on an execution plan for the interactions that form an organizational process. The paper then elaborates on the protocols that agents need to follow to perform the default interactions and arrive at a plan.

The first layer of the approach presented in this paper shows how default interactions are modeled in order to enact the plan for each interaction. The second layer of the approach shows how the protocols for default interactions are defined. The introduction of default interactions re-enforces the interaction-centric methodology proposed by TALL – because standardized mechanisms for simulation are built-in in the execution semantic of the language. The alternative is to build a very complex environment that ensures the coherence of the organizational processes. By considering the default interactions as part of the language, the burden of implementing the agent environment in an organizational simulator with TALL is lower.

References

1. Roest, G.B., Szirbik, N.B.: Escape and Intervention in Multi-Agent Systems. J. AI & Society. 24(1), 25--34 (2009)
2. Milner, R.: Action Calculi, or Syntactic Action Structures. LNCS vol. 711, pp. 105--121, Springer Verlag (1993)
3. Hoare, C.A.R.: Communicating Sequential Processes. Prentice Hall International Series in Computer Science (1985)
4. Milner, R.: A Calculus of Communicating System. LNCS, vol. 92, Springer Verlag (1980)
5. Sangiorgi, D., Walker, D.: The Pi-Calculus – A Theory of Mobile Processes, Cambridge University Press (2001)
6. Bergstra, J.A., Klop, J.W.: Process algebra – specification and verification bisimulation semantics. Mathematics and Computer Science II, CWI Monograph 4, 61-94, North-Holland, Amsterdam (1986)
7. Fokkink, Van: Introduction to Process Algebra. Texts in Theoretical Computer Science, An EATCS series. Springer Verlag (1999)
8. Agha, G.: Actors: A model of Concurrent Computation in Distributed Systems. MIT Press, Cambridge, MA, (1986)
9. Moulin, B., Rousseau D.: An approach for modelling and simulating conversations. In: Vanderveken, D., Kubo, S. (eds.) Essays in Speech Act Theory. Pragmatic and Beyond New Series, vol. 77, John Benjamins Publishing (2002)
10. Wegner, P., Arbab, F., Goldin, D., McBurney, P., Luck, M., Robertson, D.:The role of Agent Interaction in Models of Computing. Electronic Notes in Theoretical Computer Science, vol. 141, 181--198 (2005)
11. Stuit, M., Szirbik, N.B.: Modelling and Executing Complex and Dynamic Business Processes by Reification of Agent Interactions. LNAI vol. 4457, pp. 106--125. Springer, Heidelberg (2007)
12. Odell, J., Van Dyke Parunak, H., Bauer, B.: Representing Agent Interaction Protocols in UML, Agent-Oriented Software Engineering. In: Ciancarini, P., Wooldridge, M. (eds.) pp. 121—140, Springer-Verlag, Berlin (2001)
13. Bauer, B., Müller, J.P., Odell. J.: Agent UML: A Formalism for Specifying Multiagent Software Systems. LNCS vol. 1957, pp. 109—120, Springer, Heidelberg (2001)
14. Wooldrige, M., Jennings, N., Kinny. M.: The Gaia Methodology for Agent-Oriented Analysis and Design. J. Autonomous Agents and Multi-Agent Systems 3(3), 285--312. (2000)
15. Evans. R.: MESSAGE: Methodology for Engineering Systems of Software Agents, Initial Methodology. Technical Report, Eurescom (2000)
16. Caire, G., Coulier, W., Garijo, F., Gomez, J., Pavon, J., Leal, F., Chainho, P., Kearney, P., Stark, J., Evans, R., Massonet, P.: Agent Oriented Analysis Using Message/UML. LNCS vol. 2222, pp. 119—135, Springer, Heidelberg (2002)
17. Wagner, G.: The agent-object-relationship metamodel: towards a unified view of state and behaviour. J. Information Systems, 28(5), 475--504 (2003)
18. Meyer, G.G., Szirbik, N.B.: Anticipatory Alignment Mechanisms for Behavioural Learning in Multi Agent Systems. LNAI vol. 4520, pp. 325—344, Springer, Heidelberg (2007)
19. Stuit, M., Meyer, G.G.: Agent interaction modeling based on product-centric data: A formal method to improve enterprise interoperability. LNBIP vol. 25, pp. 197-219, Springer, Heidelberg (2008)
20. Mintzberg, H.: Structures in Fives, Designing Effective Organizations. Prentice Hall (1992)