From Multi-Agent to Multi-Organization Systems: Utilizing Middleware Approaches

Matthias Wester-Ebbinghaus, Daniel Moldt and Michael Köhler-Bußmeier

University of Hamburg, Department of Informatics
Vogt-Kölln-Straße 30, D–22527 Hamburg
{wester,moldt,koehler}@informatik.uni-hamburg.de

Abstract. Modern software systems share with social organizations the attributes of being large-scale, distributed, and heterogeneous systems of systems. The organizational metaphor for software engineering has particularly been adopted in the field of multi-agent systems but not entirely exploited due to an inherent lack of collective levels of action. We propagate a shift from multi-agent to multi-organization systems that we rest upon an organization theoretically inspired reference architecture. We further suggest to ultilize agent-oriented technology as a means for realization. We draw upon the wide variety of organizational modelling and middleware approaches and establish a best fit between different approaches and and requirements for different architectural levels.

1 Introduction

The characterization of modern software systems as *application landscapes* [1], *ultra-large-scale (ULS) systems* [2] or *software cities* [3] carries the comprehension of as large-scale, inherently distributed and heterogeneous *systems of systems*. The various parts of the overall system are embedded in complex and dynamic environments, where they co-evolve with various other systems (software systems, other technical systems, social systems). In this respect, the challenges in dealing with modern software systems resemble the challenges that organizations in social societies face. Despite being inherently complex, distributed and heterogeneous themselves and despite existing in potentially highly complex and dynamic environments, organizations have to draw and manage their boundaries in a way that properly supports their goals and purposes. According to Hannan and Carrol, the main capacities of organizations in comparison to other social collectivities are precisely their *durability, reliability* and *accountability* [4].

Thus, it is not surprising that multiple software engineering approaches have been brought forth in recent years that take an organization-oriented perspective on software systems. Many of these approaches are rooted within the multi-agent system (MAS) paradigm of software engineering as the underlying metaphors are already socially inspired and thus provide an ideal breeding context for organization-oriented ideas. A thorough overview of recent and current work in this field can be found in [5]. While multi-agent system research has made significant contributions to establishing an organizational metaphor for software engineering, when relating these contributions to organization theory it becomes

obvious that the true potential of the organizational metaphor is not entirely exploited. Organization theories are frequently classified along the dimension of *analysis level*. On the on hand, organizations can be regarded as systems composed of individuals for whom they provide technical and social contexts. This conception is what multi-agent system research has focussed on so far. On the other hand, Scott [6] points out that one cannot comprehend the importance of organizations in human societies if they are only regarded as contexts for individual actors. Instead, organizations are collective actors in themselves. They carry out actions, utilize resources, enter contracts and own property. Indeed, they are the primary social actors of today's society. With respect to ever larger and more complex software systems, it becomes obvious that this conception of organizations as collective actors is equally vital for software engineering. However, it has only been taken into account tentatively by multi-agent research. One might argue that this was just a matter of extrapolating the concepts for the individual level (agents as actors) to the collective level (organizations as actors). Approaches for *agentifying* multi-agent systems [7] and holonic multi-agent systems [8] pave the way for this advancement. We realize these possibilities, but at the same time insist that this transition should more heavily rely on results that organization theory as a discipline with a long tradition in investigating organizations, their internals, and their environments already has to offer.

Consequently, the aim of this paper is the provision of a software development proposal that builds upon and extends the multi-agent system approach in order to account for the true potential of the organizational metaphor. Central to our proposal is the advancement from the individual agent to the organization as a software engineering metaphor of higher granularity. In [9] Ferber advances the distinction between ACMAS (agent-centred multi-agent systems) and OCMAS (organization-centred multi-agent systems). We consider our approach as one further step in this shift of paradigm and term the systems introduced by our approach MOS (multi-organization systems). We do not claim to supplant multi-agent system philosophy by introducing an entirely new paradigm. The approach we present completely rests on an underpinning of multi-agent systems. But we do claim to take an entirely fresh look at the game of organization-oriented software architectures. In Section 2 we present the ORGAN-model (**Org**anizational **A**rchitecture **N**ets) as a "thinking model" to comprehend systems of systems under a multi-organization perspective. In Section 3 we compare three distinct organizational models for multi-agent systems that rely on middleware approaches for deployment. We investigate the prospects of the middleware philosophy with respect to the engineering of large-scale systems following ORGAN. We conclude our results in Section 4 and provide an outlook on open issues and future work.

2 Engineering Systems of Systems: The ORGAN-Model

In this section, we consider characteristics and problems of large-scale software systems and present the ORGAN-model as an organization theoretically inspired comprehension model.

2.1 Software Systems in the Large

It is necessary to go beyond the concept of size alone in order to get a grasp of the nature and effects of software systems of the category ULS system [2]. Their parts are most often derived from structures and processes of real-world systems. These include physical and mechanical systems but above all social systems. Social systems of a certain size are inherently distributed and decentralized. This condition accounts for the fact, that the accompanying software systems are not monolithic (which might also yield an enormous size) but rather *systems of systems* as characterized in [10]. The several system parts are typically not only geographically distributed but also independently acquired, deployed and useful. The overall system is not intentionally formed and developed but evolves gradually and exhibits a considerable degree of emergent behaviour.

In [2] an analysis of the problems resulting from these characteristics is carried out. The inherent geographical distribution in combination with managerial and operational independence inevitably lead to decentralization. Besides data management and operation this affects the evolution of the overall system, its control and observability. It is not possible to stop the system or to take out uniform rollouts and release changes. Instead, the functional range of the system is extended, adapted or restricted simultaneously to its operation. Thus, the evolution of the overall system is continuous rather than following well-defined phases. Governance mechanisms that assume a comprehensive knowledge of system-wide parameters, the possibility of solving conflicts uniformly, and the effective adoption of a central authority become mostly obsolete. Instead, one has to deal with decentralized control, the impossibility of global observability and assessment and difficult estimation of the effects of pertubations.

Organization-oriented MAS engineering addresses many of the just mentioned problems. In particular, they deal with autonomous and potentially heterogeneous system parts subsumed under joint superstructures. In the Section 3 we will elaborate on some related concepts. Nonetheless, one drawback of multi-agent system approaches to systems of systems engineering becomes obvious. The sheer size of the software systems addressed in this paper entails the necessity to distinguish different levels of abstraction. Degree of abstraction relates to the granularity of the system units that are studied at each respective level. The core metaphor of multi-agent system research is the individual agent, which is of rather small granularity. As mentioned in the introduction, one common distinction regarding organization theoretical studies is whether an individual or a collective level of analysis is chosen. Collective level issues are not inherently rooted in the agent paradigm. Like stated in the introduction, recent comprehensions of MAS put a stronger focus on the system/organization level and some approaches even address the topic of multiple agents acting "as a whole" and thus exhibiting corporate agency. These efforts narrow the gap between the agent paradigm and the demand for collective level perspectives. However, there remains the risk of a mismatch between the applied core metaphor and the required conceptual level, at which application problems have to be addressed.

For this reason, we advocate a temporary departure from the agent paradigm at this point. Instead, we introduce the ORGAN reference model for comprehending systems of systems under an organization-oriented perspective.

2.2 Universal Model of an Open, Controlled System Unit

The underpinning of ORGAN is the universal model of an open and controlled system unit from Figure 1 that is applied at *all* system levels.[1] Three types

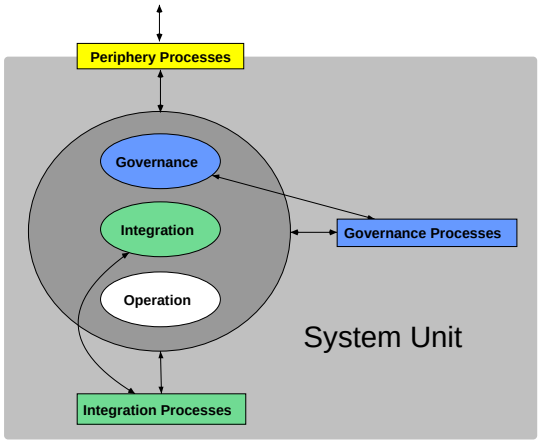

Fig. 1. Universal model of an open, controlled system unit

of internal system units and system processes are distinguished respectively. Internal system units are categorized as *operational units, integration units* and *governance units*.[2] Integration units together with operational units represent the "here and now" of the system unit in focus. The operational units are so to say the intrinsic units and carry out the system's primary activities. They are dependent on the integration units which offer a *technical frame* via intermediary,

[1] This model has an underlying *reference net* semantics [11]. Reference nets carry some extensions compared to ordinary coloured Petri nets. They implement nets-within-nets concept where a surrounding net can have references to other nets as tokens. Synchronous channels allow for bi-directional synchronous communication between net instances. In [12], we show how the approach of modelling system units with reference nets allows for straightforward prototyping and simulation.

[2] It is important to note that operation, integration and governance are in the first place *analytical aspects* of system units. However, they are carried and backed up by certain (individual or collective) actors, hence the classification of internal system units. Although analytically distinct, the three aspects are intrinsically interwoven and interdependent. The three sets of internal system units do not even have to be disjoint. Instead, particular system units might fulfill multiple analytical roles.

regulation, and optimization services in the course of integration processes. The governance units represent the "there and then" of the system unit in focus. They offer a *strategical frame* via goal/strategy setting, boundary management, and transmitting their decisions to the other internal system units in the course of governance processes.

Each system unit is a *Janus-faced* entity. It embeds system units as internal system units and is itself embedded in other system units as one of their internal system units. Thus, besides the already mentioned internal frames (technical and strategical) each system unit in focus is *externally framed* by surrounding system units to which the system unit in focus (by means of its internal system units) relates via periphery processes.

To conclude, we take a recursive, self-similar nesting approach, borrowed from Koestler's concept of a *holon* [13] that we extend with a generic reference model for control structures at each level. We arrive at a modular approach to comprehend systems of systems. Each system part may be regarded under a platform perspective and under a corporate agency perspective. This provides a conceptual basis to systematically study and implement different modes of coupling, both horizontally and vertically.

2.3 Reference Architecture

With respect to software architectures, a selective distinction of different system levels has to be carried out. Here, we advance the ORGAN reference architecture for multi-organization systems.

Overview The core unit now is the organization instead of the agent. This directly leads to three mandatory architectural levels, the organization itself, its internals and its environment. As an organization might have different conceptual environments (e.g. different domains within which it operates) and the architecture targets at the inclusion of multiple organizations we have to deal with multiple environments. Thus, the need for fourth architectural level that acts as a system closure arises.

This distinction of architectural levels resulting from rather technical considerations is confirmed by analysis levels that organizational theories target at according to Scott's classification [6]. At the *socio-psychological level* the internals of an organization in terms of relationships between its individual members are studied. The *organization structure level* introduces the perspective on an organization as a holistic, identifiable entity. It studies structures and processes that characterize an organization in terms of its parts (departments, teams) and its analytical components (specialization, communication network, hierarchy). The *ecological level* focuses on characteristics and actions of organizations as corporate actors operating in an even wider network of relations. For this level, further distinctions are possible. Among them, the level of an *organzational field* bears the most comprehensive concept of an organization's environment. It consists of organizations that, in the aggregate, constitute a recognized area of institutional life (key suppliers, consumers, regulatory agencies, etc.). Finally, the *society* integrates all fields under a common body of law.

Consequently, four types of *organizational units* as specific forms of the universal model of system units from Figure 1 follow for the ORGAN-architecture: departments, organizations, organizational fields and the society. Figure 2 shows an illustration. In the remainder of this section, we differentiate the four types ac-

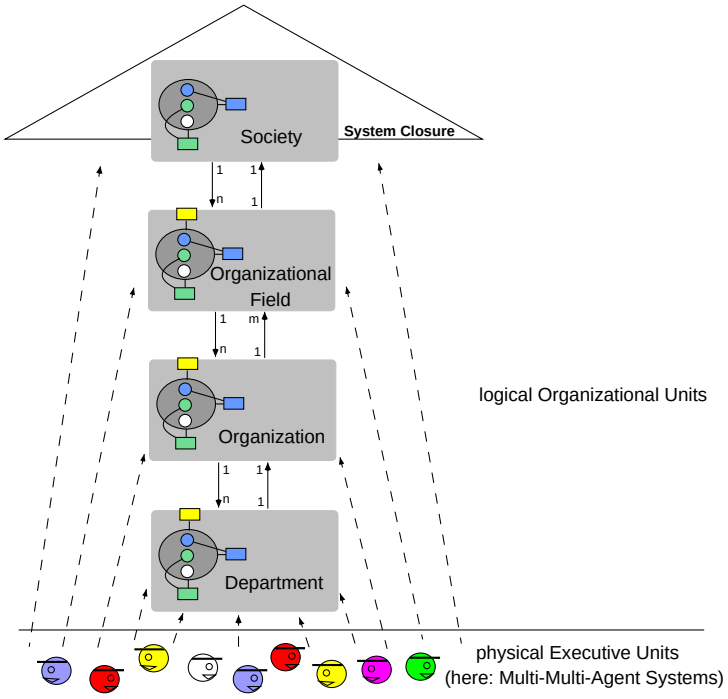

Fig. 2. ORGAN reference architecture

cording to the characteristics that operation, integration and governance take on respectively (cf. [14] for further details). In addition, we investigate the architectural role of each organizational unit under a software engineering perspective.

Society There exists one organizational unit of the type society (but see the conclusion). The society represents the upper closure of the system. Each software system serves a certain purpose (or certain purposes). This is mirrored by the architectural role of the society. The society supports a specific extract of reality from which the purpose of the overall system is derived. The *operational* aspect is embodied by different scopes of the overall system (e.g. healtcare, power supply, e-commerce). For each system scope the society embeds one organizational field. These fields also make up the *integrational* aspect by managing transitions and interactions between each other. *Governance* is taken care of by specific fields that constitute the society's government. It specifies system laws,

some of which have a system-wide validation and some of which only hold for certain fields. For this purpose the society's government guards over these laws and utilizes governance structures at the field level as caretakers for these laws.

Organizational Field Each organizational field provides a consistent and largely self-contained picture of a particular part of the overall system ("living space"). Fields represent the immediate environments for organizations. Participating Organizations as the *operational* part of a specific field have some functional interest in common and face each other as discrete and co-equal entities (cooperating or competing out of self-interest). However, they are embedded in the common environmental frame of the field. Here we can distinguish the material-resource features as an *integrational* aspect from the institutional features with both *integrational* and *governance* aspects. Material-resource features capture factors directly related to the demand, supply and exchange of products and services on the field. Thus, they are to a great part inherently constituted by the participating organizations themselves, but also encompass organizations that enable and mediate organizational interchange in the first place or offer auxilliary services (e.g. industry associations, market organizers, assurance companies, banks, consultant offices). Material-resource features always rest on institutional foundations that are made of practices and symbolic constructions constituting the organizing principles of the field. Thus, we find organizations for setting (e.g. regulatory agencies, professional associations) and implementing (e.g. market oversight or facilities for licensing and certification) the institutional logics of a field. These institutional logics are available for participants to individually elaborate, which fosters a more efficient design of organizations.

Organization Just like agents in agent-oriented software engineering, organizations in the context of the ORGAN-architecture are the central modelling units that determine, at which level of abstraction the whole system is regarded. Organizations operate on one or more organizational fields, depending on the variety of domains, to which they relate. Organizations are composed of departments, which mirror the complexity that the organizations face in their environments. Different needs and functions of the organization with regard to their environmental embedding are mapped onto different departments. Contrary to organizational fields that host distinct and co-equal entities, the departments of an organization are dependent on the unique organization they belong to and exist on their behalf. They are fused into a joint superstructure under a joint strategy to pursue and achieve joint goals. Many departments fulfill an *operational* as well as an *integrational* role. One particular department might act as an integrational unit for departments that it groups according to the organization's superstructure. At the same time, it might act as an operational part towards another department to whose grouping it belongs. The superstructure is typically hierarchical in nature but may be augmented by various vertical ties. Finally, the *governance* aspect is taken care of by departments that constitute the dominant coalition (including at least high-ranked managers from within the organization, but potentially encompassing further actors from within and even without the organization) of the organization. It sets the organization's goals and strategies.

Department The departments are the actuators of the overall system. They represent the final implementation means for all higher-level system activities. Each department exists on behalf of one unique organization and this organization determines the department's characteristics concerning structure and processes. Here, a continuum of possibilities opens up, from rather *bureaucratic* structures (high degree of standardization) to rather *organic* structures (low degree of standardization). Each department is *governed* by its management and its *operational* members are *integrated* by their respective positions in the department's context.[3]

To conclude, the ORGAN-model for organization-oriented software architectures provides a conceptual thinking model for large-scale software systems. Contrary to an agent-oriented perspective, collective levels of action are inherent to a truly organization-oriented perspective. However, when it comes to actually realizing multi-organization systems according to the ORGAN-model, we are of the opinion that current MAS technology provides an ideal starting point. This opinion is illustrated in Figure 2 where the organizational units are considered as *logical* constructs that have to be incarnated by means of *physical* executive units, which is accompanied by a *change in paradigm*. In the following Section 3 we elaborate on our proposal to utilize and combine MAS (middleware) approaches in order to realize multi-organization systems.

3 Utilizing Multi-Agent Middleware Approaches

Organization-oriented approaches to multi-agent system design employ the mechanism of *formalization* borrowed from social organizations. Formalization in this context refers to the extent, to which expectations on behaviour are explicitly and precisely specified, and to the extent, to which these specifications are independent from the particular occupants of social positions [6]. In this respect, rationality resides in the social structure itself, not in the individual participants. Adopting this principle for multi-agent system engineering allows for separation of concerns. Organizational specifications and the agents that fill these specifications with life may be designed separately. The aim is to combine local agent autonomy with the assurance of global system characteristics.

Some approaches carry separation of concerns to the implementation level. Instead of just resting "in the heads" of the participating domain agents, organizational specifications are encapsulated and managed by an explicit middleware and thus software technically *reified*. This allows arbitrarily heterogeneous agents to participate in the organization as the middleware acts as an intermediary. Furthermore, it is extremely useful in open multi-agent environments where agents

[3] Having an immaterial concept like an organizational position as an integration unit at the department level may seem odd at first glance. However, formalizing position characteristics for social organizations is first step in reifying positions. For software systems, this may be carried even further as will be shown in Section 3 where we consider different middleware approaches for organization-oriented MAS engineering.

belonging to different stakeholders continuously enter and leave and the organizational specifications have to be buffered against potentially harmful influences.

The specific characteristics of a given middleware approach depend on the characteristics of the organizational specifications that are to be supported. Here, we examine three distinct approaches. Afterwards, we analyze how the benefit of separation of concerns can even be extended with respect to not only separating organizations and domain agents but also different system levels of software systems in the large. We specifically consider the levels introduced by the ORGAN-architecture from Section 2.

3.1 Middleware Approaches

To distinguish different modelling approaches we adopt the approach taken in [15] where distinctions are made based on different *organizational dimensions* that are supported.

$\mathcal{M}$OISE$^+$/$\mathcal{S}$-$\mathcal{M}$OISE$^+$ The $\mathcal{M}$OISE$^+$ modelling language [16] incorporates a structural, functional and deontic dimension. In the *structural dimension*, roles and groups are specified. Roles are related to one another via inheritance relationships. Groups consist of a set of roles where for each role the minimal and maximal cardinality is specified. In addition to inheritance, additional links (compatibility, authority, communication, acquaintance) with either an intra-group or inter-group scope may exists between roles.

The *functional dimension* consists of a set of social schemes. A scheme is a goal decomposition tree where the root is the initial organizational goal the scheme targets at. Groups may be linked to schemes. Roles belonging to the group are then assigned to coherent sets of the scheme's goals (so called missions) via permission or obligation links in the *deontic dimension*.

$\mathcal{S}$-$\mathcal{M}$OISE$^+$ [17] is the middleware for supporting $\mathcal{M}$OISE$^+$ specifications in open multi-agent systems. Figure 3 gives an overview of its architecture. Domain

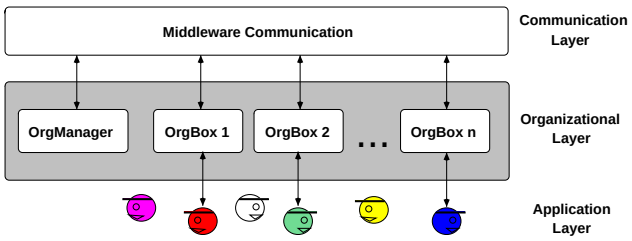

Fig. 3. $\mathcal{S}$-$\mathcal{M}$OISE$^+$ middleware approach (adapted from [17])

agents connect to the organization via proxys called *OrgBoxes* that offer an API representing the agents' possibilities to act as members of the organization. All requests for changes in the state of the organization (e.g. role adoption, group

creation, mission commitment) are routed to a central *OrgManager*. Requests are only fulfilled if no organizational constraints are violated. The OrgManager can also act proactively, for example by informing agents (mediated by their OrgBoxes) of missions they are enforced to commit to or goals that have become ready to pursue. Agents have access to the organizational specification and are free to interpret it to optimize their organization-aware reasoning.

ISLANDER/AMELIE ISLANDER [18] introduces a conceptual shift from organizations to institutions. Instead of being constructive in terms of goal/task trees ($\mathcal{M}$OISE$^+$/SONAR) to achieve certain organizational objectives in a divide-and-conquer style, ISLANDER focuses on the regulative character of institutions.

ISLANDER supports a strutural, dialogical/interactional, and a normative dimension. The *structural dimension* differs from the $\mathcal{M}$OISE$^+$ structural dimension in that it does not relate roles and groups but so called scenes. Scenes represent the *interactional dimension*. A scene is basically a collection of roles in interaction with each other following a well-defined interaction protocol. Each scene embodies a largely self-contained collective activity of the overall system. In the structural dimension, relationships among scenes are established by a so called performative structure. It specifies the network between scenes and defines transitions between scenes. Transitions define which agents playing which role under which circumstances can move from one scene to another and whether new instances of scenes have to be brought up upon firing transitions.

An additional *normative dimension* specifies consequences of agent actions. A norm defines, which obligations hold after certain communicative acts have (or have not) been uttered in certain scenes and certain side conditions hold.

AMELIE [19] is the middleware for supporting ISLANDER specifications. Figure 4 gives an overview of its architecture. Just like in $\mathcal{S}$-$\mathcal{M}$OISE$^+$, agents do

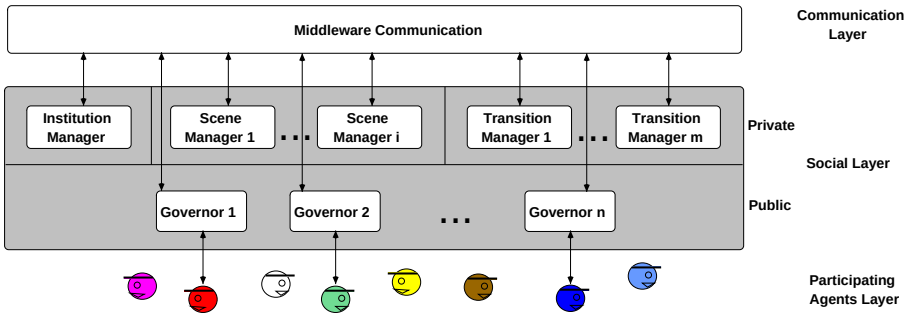

Fig. 4. AMELIE middleware approach (adapted from [19])

not interact directly but connect to middleware mediators, in the case of AMELIE these are the *governors*. Contrary to $\mathcal{S}$-$\mathcal{M}$OISE$^+$ there exist different roles in managing the institution. An *institution manager* is in charge of the institution

as a whole. I starts the institution, authorises agents to enter and manages the creation of new scene executions. Each scene execution is managed by a separate *scene manager* overseeing the execution according to the associated scene protocol. *Transition managers* are in charge of managing agent transitions between scenes. The different management parts communicate with each other and with the governors. Each governor manages multiple conversations with its respective connected agent, one conversation for each scene and transition participation. In addition, it keeps track of the norms that concern its associated agent.

SONAR SONAR [20] is a mathematical model of multi-agent organizations based on Petri nets. It has a structural, functional, and interactional dimension. However, *structural and functional dimension* are inseparable. The functional part consists of a Petri net, where each place models a task and each transition models the execution of the task associated with the unique place in the transition's preset. Task executions may introduce subtasks, which results in a multi-tree structure with multiple roots associated with core tasks. This functional specification is enriched with structural information by partitioning the Petri net by means of so called organizational positions. Consequently, each position is responsible for the execution of some tasks and possibly permitted to delegate (sub-)tasks to other positions.

The *interactional dimension* comes into being by relating each place with a set of roles and each transition with an interaction protocol. Thus, tasks correspond to the implementation of roles and task executions to the interactions that have to be carried out between these roles. Subtasks correspond to the refinement of roles into subroles whose implementation is further delegated.

The SONAR middleware approach is illustrated in Figure 5. It differs from

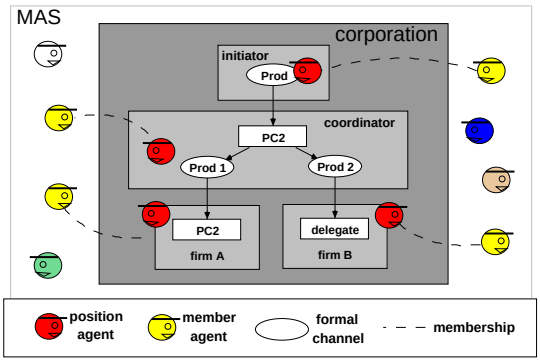

Fig. 5. SONAR middleware approach (adapted from [21])

the two already presented ones. It does introduce a middleware layer, but this layer is not *physically* distinguishable from the application layer. Instead, the layer *logically* consists of *position agents*, one for each organizational position.

Position agents are launched on behalf of the organization. Domain agents have to connect to them in order to act as members of the organization. Position agents mediate interactions and take care that the organizational specifications are honoured. But contrary to $\mathcal{S}$-$\mathcal{M}$OISE$^+$ and AMELIE, SONAR advocates complete distribution. Each position agent only knows its local context ("upwards" and "downwards" delegation partners, own tasks, own task executions, associated roles and interactions patterns). There is no central facility (or some central facilities) that keeps a global picture and exploits centralized coordination.

3.2 Multiple Layers of Middleware

The benefit of separation of concerns fostered by middleware approaches basically aims at separating domain agent and organizational concerns. However, the "agent neutrality" of middleware layers can be carried forward to separate the engineering of different system levels in the context of large-scale software systems. Figure 6 shows an illustration of this idea. To realize it, both organiza-

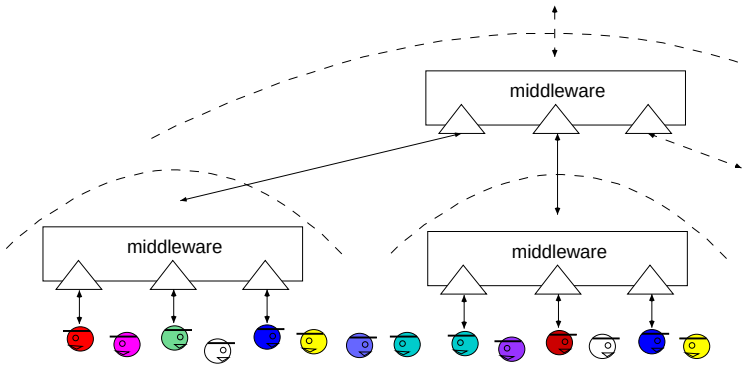

Fig. 6. Iterating middleware layers

tional modelling languages and their supporting middleware frameworks would have to incorporate a *Janus-faced* character. In the case of "looking downwards", not much changes. In case of "looking upwards", organizational modelling languages would have to take into consideration processes that are not entirely enclosed by the modelled system in focus. The accompanying middleware implementations need to be able to connect to other middleware layers themselves and to mediate activity not only horizontally but also vertically.

The concepts of a Janus-faced character and periphery processes were also stressed in the context of the ORGAN model in Section 2.2. Thus, we consider the approach from Figure 6 as a promising realization strategy to engineer software systems in the large according to ORGAN. The whole idea of recursive, self-similar nesting is by no means new to multi-agent system research and is for example incorporated by the concept of *holonic multi-agent systems* [8]. In our opinion,

nested middleware layers foster such an approach in the context of large-scale systems where there exists a strong need for the modular design, creation, and maintenance of distinct system parts.[4]

3.3 Pooling of Competencies

Like stated in Section 2, different levels of a system call for different modes and paradigms (not only degrees) of coupling and organizing. Here we may avail ourselves of the broad spectrum of already existing organizational/institutional multi-agent approaches supported by middleware. We exemplify this potential by relating the three approaches from Section 3.1 to the different layers of the ORGAN-architecture from Section 2.2. A primary distinction was already mentioned. While $\mathcal{M}$OISE$^+$ and SONAR are rather constructive in terms of achieving certain organizational purposes, ISLANDER is rather declarative in focussing on regulations. For this reason, we consider ISLANDER ideally suited for the level of organizational fields and the other two approaches best suited for the level of departments and organizations.

Department Level For the department level, a high degree of detail is necessary. Ultimately, departments are both source and sink for all activities. It is not only necessary to model activities but also under which circumstances and conditions they come into being. $\mathcal{M}$OISE$^+$ is very rich when it comes to model relationships between actors in terms of role inheritance, various types of additional links (compatibility, authority, acquaintance, communication) between roles and clustering of roles into groups. For this reason, $\mathcal{M}$OISE$^+$ is ideally suited for the level of departments. Even the lack of an interactional dimension in some way suits the department level. Social schemes might be considered as abstract programs but how these programs are actually followed for particular instances might be open to mutual adjustment (only possible if all participating roles shared mutual communication links).

In addition, the centralized management of the $\mathcal{S}$-$\mathcal{M}$OISE$^+$ middleware fits the department level. Departments exhibit the notion of locality in the sense that the network of interdependencies is quite tight and highly intermeshed. Thus obtaining a global picture of what is going on is very useful. The drawbacks of centralized solutions like risk of bottleneck and high response times should pose no severe problems as number of participants as well as network distances are typically at small or medium scale for departments.

Organization Level SONAR models on the other hand are more abstract and much better suited for the organizational level where not individuals but departments are related to one another. The core unit of abstraction in SONAR

[4]The middleware approaches from Section 3.1 tend to establish anonymity between domain actors. As the explanations from Section 2.2 should have made clear, this conception is not adequate for software systems according to the ORGAN-model where there may exist a quite high degree of penetration and visibility between certain organizational units. It is a matter of designing organizational/institutional processes in an appropriate way to establish the desired degrees of coupling and acquaintance.

is the position rather than the role. Each position is unique like it is typically the case for departments in an organization. The level of detail $\mathcal{M}$OISE$^+$ offers for relating roles to one another (e.g. inheritance, compatibility, cardinality) is not necessary when relating departments instead of individuals. The most frequently adopted view when regarding organizations nowadays is a business process perspective. Entities related to those processes are characterized by abstract service interfaces that define which functionality each respective entity contributes. SONAR exactly offers such a process-centric perspective. Each position is responsible for the execution of several tasks and may delegate (sub-)tasks to other positions. This can be considered as the abstract service description of each position ("offers", "uses") and clearly defines what each position has to contribute to business processes. An explicit interactional dimension is vital as business processes may relate entities that are quite depart from one another (locally as well as conceptually), which mostly prohibits mutual adjustment. Starting business processes and actually supplying services is a matter of the underlying $\mathcal{M}$OISE$^+$ departments.

The completely distributed middleware approach taken by SONAR perfectly matches organizations as large and widely distributed entities. Maintaining a centralized comprehensive picture of a whole organization would be very costly and induce an enormous information overhead. It is not even necessary in the first place. Networks at the organizational level are less intermeshed than it is the case for members at the department level. Organizations are better characterized by (partly overlapping) clusters of frequent interaction.

Organizational Field and Society Level Ascending to the level of organizational fields, there no longer exist explicit notions of joint goals or strategies. Instead, organizations cooperate and compete as co-equals in order to serve their respective distinct purposes. At the same time, governance structures enforce common institutional logics that field participants have to adhere to. Here lies the core competency of ISLANDER. It specifies an environmental framework that regulates the behaviour of participants by specifying what one is allowed, forbidden or obliged to do in certain institutional contexts. The inherent goal is to let participants pursue their respective goals, but on the basis of well-defined practices and symbolic constructs. ISLANDER's performative structure and scene protocols are open for participants to elaborate in order to fit their individual behaviour into the institution. With ISLANDER applied to the field level, SONAR organizations are to adjust their business process specifications in a way to act in and travel between scenes.

Each organizational field might be represented by one single scene or by a set of scenes. As ISLANDER's performative structure provides transitions between scenes it scales in order to include an arbitrary number of fields and thus also addresses the level of society.

Fortunately, AMELIE offers a distributed middleware approach that is needed for the field and society level. However, scene managers might become overloaded if a field was embodied by one or just a few scenes.

4 Conclusion and Outlook

We have presented a proposal to progress from multi-agent to multi-organization systems, which we consider a necessary transition in the face of software systems of ever growing size and complexity. We see one main contribution in the explicit identification and differentiation of collective level aspects in our conceptual framework ORGAN. It features different modes of collective action that are adequate for different contexts. The accompanying concepts and principles are deeply rooted in organization theory. The rationale behind this approach is to learn from the adaptivity, robustness, scalability and reflexiveness of social (multi-)organization systems and to translate their building principles in effective information technology.

In addition, we have made a suggestion of how to utilize agent-oriented technology as a realization means. One particular point is to take advantage not only of current middleware implementations but also of the variety of underlying modelling approaches in order to establish a most adequate fit between them and the requirements of different system levels. While this proposal is somewhat preliminary and the current state-of-the-art of the presented middleware approaches does not match the requirements of our nested middleware proposal, we consider it a promising vantage point.

Another future issue is to soften the 4-layer restriction of ORGAN. Following a multi-perspective approach, an organizational unit might be able to occupy multiple architectural roles in multiple instances at the same time. This puts stronger emphasis on the *relations* between units instead of the units themselves. Depending on relation, units may take on different manifestations in terms of architectural roles. This allows for example for federations (organizations embedded in organizations as departments), subfields (fields embedded in fields as organizations) or societies of societies (societies embedded in societies as fields). The universal basis of all organizational units in form of the model of a system unit from Figure 1 fosters such a multi-perspective conception.

References

1. Lankes, J., Matthes, F., Wittenburg, A.: Softwarekartographie: Systematische Darstellung von Anwendungslandschaften. Wirtschaftsinformatik 2005 (2005)
2. Northrop, L.: Ultra-Large-Scale Systems: The Software Challenge of the Future. Software Engineering Institute, Carnegie Mellon (2006)
3. Hess, A., Humm, B., Voss, M., Engels, G.: Structuring software cities - a multidimensional approach. In: Proceedings of the 11th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2007). (2007) 122–129
4. Hannan, M., Carroll, G.: An introduction to organizational ecology. In: Organizations in Industry: Strategy, Structure and Selection. New York: Oxford University Press (1995) 17–31
5. Wester-Ebbinghaus, M., Moldt, D., Reese, C., Markwardt, K.: Towards Organization–Oriented Software Engineering. In: Software Engineering Konferenz 2007 in Hamburg: SE'07 Proceedings. Volume 105 of LNI., GI (2007) 205–217

6. Scott, W.R.: Organizations: Rational, Natural and Open Systems. Prentice Hall (2003)
7. Rölke, H.: Modellierung von Agenten und Multiagentensystemen. Logos Verlag Berlin (2004)
8. Fischer, K., Schillo, M., Siekmann, J.: Holonic multiagent systems: A foundation for the organization of multiagent systems. In: Holonic and Multi-Agent Systems for Manufacturing, First International Conference on Industrial Applications of Holonic and Multi-Agent Systems (HoloMAS). Volume 2744 of Lecture Notes in Computer Science., Springer Verlag (2003) 71–80
9. Ferber, J., Gutknecht, O., Michel, F.: From agents to organizations: an organizational view of multi-agent systems. In: Agent-Oriented Software Engineering IV, 4th International Workshop, AOSE 2003. Volume 2935 of Lecture Notes in Computer Science., Springer Verlag (2003)
10. Maier, M.: Architecturing principles for systems-of-systems. Systems Engineering **1**(4) (1999) 267–284
11. Kummer, O.: Referenznetze. Logos Verlag, Berlin (2002)
12. Wester-Ebbinghaus, M., Moldt, D.: A janus-faced net component for the prototyping of open systems. To appear, accepted paper for the 15th Workshop on Algorithmen und Werkzeuge für Petrinetze (AWPN'2008) (2008)
13. Koestler, A.: The Ghost in the Machine. Henry Regnery Co. (1967)
14. Wester-Ebbinghaus, M., Moldt, D.: Structure in threes: Modelling organization-oriented software architectures built upon multi-agent systems. In: Proceedings of the 7th International Conference an Autonomous Agents and Multi-Agent Systems (AAMAS'2008). (2008) 1307–1311
15. Boissier, O., Hübner, J., Sichman, J.S.: Organization oriented programming: From closed to open systems. In: Proceedings of the Seventh International Workshop on Engineering Societies in the Agents World (EASW 2006). (2006)
16. Hübner, J.F., Sichman, J.S., Boissier, O.: A model for the structural, functional and deontic specification of organizations in multiagent systems. In: Proceedings of the 16th Brazilian Symposium on Artificial Intelligence (SBIA'02). Volume 2507 of Lecture Notes in Artificial Intelligence., Springer Verlag (2002)
17. Hübner, J.F., Sichman, J.S., Boissier, O.: S-moise: A middleware for developing organised multi-agent systems. In: International Workshop on Organizations in Multi-Agent Systems: From Organizations to Organization-Oriented Programming (OOOP 2005). (2005) 107–120
18. Esteva, M., Padget, J., Sierra, C.: Formalizing a language for institutions and norms. In: Pre-Proceedings of the 8th International Workshop on Agent Theories, Architectures and Languages (ATAL-2001). (2001) 106–119
19. Esteva, M., Rodriguez-Aguilar, J., Rosell, B., Arcos, J.: Ameli: An agent-based middleware for electronic institutions. In: Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2004). (2004) 236–243
20. Köhler, M.: A formal model of multi-agent organisations. Fundamenta Informaticae **79**(3–4) (2006) 415–430
21. Köhler, M., Wester-Ebbinghaus, M.: Closing the gap between organizational models and multi-agent system deployment. In: Multi-Agent Systems and Applications V. Volume 4696 of Lecture Notes in Computer Science., Springer-Verlag (2007) 307–309