Modelling Multi-Agent Systems with Organizations in Mind

Matthias Wester-Ebbinghaus and Daniel Moldt

University of Hamburg, Department of Informatics
Vogt-Kölln-Straße 30, D-22527 Hamburg, Germany

Abstract. Software systems are subject to increasing complexity and in need of efficient structuring. Multi-agent system research has come up with approaches for an organization-oriented comprehension of software systems. However, when it comes to the collective level of organizational analysis, multi-agent system technology lacks clear development concepts. To overcome this problem while preserving the earnings of the agent-oriented approach, this paper propagates a shift in perspective from the individual agent to the organization as the core metaphor of software engineering targeting at very large systems. According to different levels of analysis drawn from organization theory, different types of organizational units are incorporated into a reference architecture for organization-oriented software systems.

1 Introduction

Modern software systems are subject to ever increasing size and complexity. Within the IT community the expectation begins to form that the sheer size and speed of growth of these systems render most traditional software engineering approaches (relying on a top-down design, expecting comprehensive knowledge of relevant system-wide parameters, based on the possibility of applying a central control facility) inept [16]. It is these software systems in the large that the article at hand is devoted to.

Multi-agent systems as a software engineering paradigm are one candidate to provide solutions for this kind of systems. Exemplary in [11], Jennings argues that multi-agent systems are very well suited for the realization of three particularly important techniques for handling complex software, namely *decomposition, abstraction, and organization*. At the same time however, Jennings calls for a *social level view* on multi-agent systems in order to deal with the difficulties that agent autonomy and sophistication in combination pose on the prediction of the overall system behaviour by leading to a considerable scope of emergence.

In subsequent years until today various approaches have been brought forth that are in line with this request by taking the perspective on a multi-agent system as an *organization* (an overview of recent and current work can be found in [19]). Noteworthy, the rationale for adopting an organization-oriented perspective throughout the approaches mostly refer to the very same features that Hannan and Carroll identify as the main capacities of organizations in human societies: Organizations are *durable, reliable* and *accountable* [10].[1] In this respect, organization-oriented approaches to multi-agent system engineering seek to combine local agent autonomy with the assurance of global system characteristics by imposing ”organizational facts” onto the system. Boissier [3] identifies different organizational dimensions (with structural, functional and interactional dimensions as the most prominent ones) that receive varying emphasis depending on the particular approach.

However, when relating multi-agent system approaches to organization theory it becomes obvious that the true potential of the organizational metaphor is not entirely exploited. Multi-agent system research so far has mainly focussed on the conception of organizations as contexts for individual agents. As we have analyzed in [19], the importance of organizations as corporate actors that Scott [18] stresses for more global (ecological) levels of analysis has been largely neglected.

Consequently, the long term goal of our work is the provision of a software development approach that builds upon and extends the multi-agent system approach in order to account for the true potential of the organizational metaphor. In this paper we supplement this goal by proposing an abstract reference architecture for organization-oriented software systems. Ferber [8] 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 from agent- to organization-orientation and term the systems introduced by our approach MOS (multi-organization systems).

In Section 2 we present our approach of modelling open and controlled system units. We utilize the introduced universal scheme to propose a reference architecture for multi-organization systems composed of concrete organizational units in Section 3. We conclude our results and provide an outlook to future work in Section 4.

2 Open System Modelling

The introduction should have made clear that the software systems addressed here are to be comprehended as *systems of systems*. To arrive at an illustrative modelling approach despite the inherent complexity of the systems in focus we adopt the modular view of each system as a *unit* that maintains relationships to other system units. As a preliminary step to dealing with particular types of system units in our organization-oriented architecture, this section refers to system theory in order to introduce a general model.

2.1 Basic System Unit

Most of the important entities studied by scientists – nuclear particles, atoms, molecules, cells, organs, organisms, communities, organizations, societies, solar systems – come under the category of a system [2]. Consequently, the characterization of a system follows quite abstract as *an assemblage or combination of parts whose relations make them interdependent*.

[1] It will not be discussed in this paper that each of these capacities is a double-edged sword and not all organizations measure up to them.

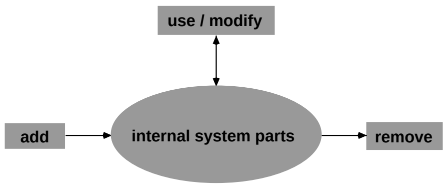

Fig. 1. An Abstract View on System Units.

Based on this abstract characterization, Figure 1 shows a general model of system units.[2]

The evolution of the overall system unit solely depends on its internal parts, whether new parts are added (add), former parts are removed (remove) or current parts are used and potentially modified (modify / use). The details of these operations and how they come into being depend on the characteristics of a particular system.

2.2 Recursive Nesting

The basic model of system units of the former subsection emphasizes the similarities of all types of systems. However, there exist of course substantial differences between particular systems. Exemplary, Boulding [4] presents a typology of systems that advances from physical over biological to social systems. Along the way, each successive system becomes progressively more complex, more loosely coupled, more dependent on information flows, more capable of self-maintenance and renewal, more able to grow and change, and more open to the environment.

Especially the openness to the environment is of particular importance regarding the software systems this paper addresses. Systems of systems implicate a network of relations where each system (to different degrees) relies on the services and resources of other systems. Interactions with the environment and throughput of external resources are regarded as crucial for the functioning and self-maintenance of open systems [17, 5]. Nevertheless, open systems have boundaries and spend energies to maintain them. Consequently, one can identify the *twin properties of open systems* as consisting of two basic (and opposing) sets of system processes [5]. The term *morphostasis* refers to those processes that tend to preserve or maintain a system's given form, structure, or state. The term *morphogenesis* refers to processes that elaborate or change the system. While both are not exclusive to open systems they receive a special emphasis. In adapting to the external environment, open systems typically become more differentiated in form and more elaborate in structure.

It is not very helpful to regard the environment of an open system as simply ”everything else”. It suggests itself to comprehend the environment again as a system (or multiple systems) and thus to arrive at a recursive understanding of open system models as for example explicated by Koestler's concept of a *holon* [12] and in Beer's recursive model of *viable systems* [1]. Each open system is characterized as ”Janus-faced” with

[2] The model has a coloured Petri nets semantics, cf. [9].

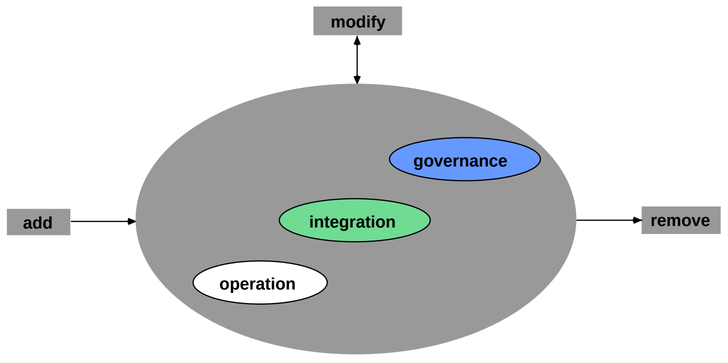

Fig. 2. System Unit with Coloured Internal Parts.

an *inner eye* at the internal systems and an *outer eye* at the surrounding system (or systems). Here, our perspective on systems as units leads to an illustrative understanding. The relation of open systems to their environments is traced back to the (not necessarily unique or disjoint) nesting of system units, the embedding of system units inside other system units.

We use this short summary of system theoretical inspirations to refine our model of system units. The first step is a *colouring* of the internal parts of a system unit as shown in Figure 2.[3] The internal parts are now explicitly to be regarded as system units themselves. The colouring is no partitioning, the different sets of internal units need not be disjoint. It is more of a conceptual classification according to function.

The *operational units* are those parts of the system in focus that undertake the system's primary activities. In a manufacturing setting they would be the production units, teams of people, and machines that actually do the manufacturing. In a complex production organization they would include manufacturing, distribution, and warehousing. Basically, the operational units are the intrinsic parts of the system.

The *integrational units* see to it that the singular operational units are integrated into a joint system in the first place. They define the means by which the operational units may participate in the system and regulate their activities. One example is the nervous system of the human organism that connects the muscles and organs (being operational units). In another setting, they embody hierarchical planning and performance control systems of an enterprise.

Governance units are responsible for certain system processes and structures being in place, to ensure the adherence to system laws, and to maintain mechanisms for control and coercion. One example is the brain of the human organism that oversees the complex of muscles and organs and tries to optimze them. It also establishes a connection to the environment through its senses. It plans, projects and develops an identity.

[3] The transitions connecting to the outer circle are just short forms that include all three cases for the inner circles respectively. To obtain a well-formed Petri net model the short forms have to be resolved, resulting in a total of nine transitions for Figure 2.

Another example is the board of directors of an enterprise that sets the goals and strategies, determines the budget and establishes connections to core business partners.

A clear separation between the different types of internal system units is not always possible just as it is not always clear where to locate morphostasis and morphogenesis. For example, in one system the purpose of the governance units might be to just preserve and maintain a set of largely fixed system laws. In another case, the government units might be a continuous source of major renewal.

Generally speaking, the distinction serves to carry out separation of concerns in two ways. Firstly, the administrative units are distinguished from the operational units that they embed. Secondly, technical embedding (via the integrational units) is distinguished from strategical embedding (via the governance units).

2.3 Structure in Threes

The overall behaviour of a system unit manifests in various system processes that shape and evolve the system. In order to study these processes exhaustively, we identify no less than 27 case distinctions that stem from three orthogonal dimensions with three values respectively.

– *Operation:* The three basic operations that a system process may relate to are *add*, *remove*, and *use/modify*.
– *Direction:* System processes may impact the system from three conceptual directions. The operational units (enabled by the integrational units) impact the system *from below*. The maintenance units impact the system *at the same level*. Finally, surrounding system units may impact the system in focus *from above*.
– *Affected internal system unit:* Each system process may involve all three kinds of system units, *operation*, *integration*, and *governance*.

The details of each case and whether it is of relevance at all depend on a particular system. Consequently, there is no point in addressing each individual case for our general model of system units. Instead, we provide coarsened models for the three directions that summarize multiple cases of the other two dimensions.

Figures 3 - 5 show the influences on the system from below, at the same level, and from above through the integration of operational units, controlling activities by governance units and peripheral connections to surrounding system units respectively.

To obtain a modelling for each of the 27 cases the first step is to refine the transitions that connect to the outer circle of the system unit in focus. In addition, the three cases shown here are somewhat ”pure” cases. In particular systems they are typically merged.

3 Reference Architecture

The refined model of open system units still does not offer a meaningful architectural model for IT systems. It lacks differentiation for particular perspectives. The central concern of this section (and the overall paper at hand) is to advance a proposal for software architectures in the large based on the universal scheme of open system units and their embedding inside each other.

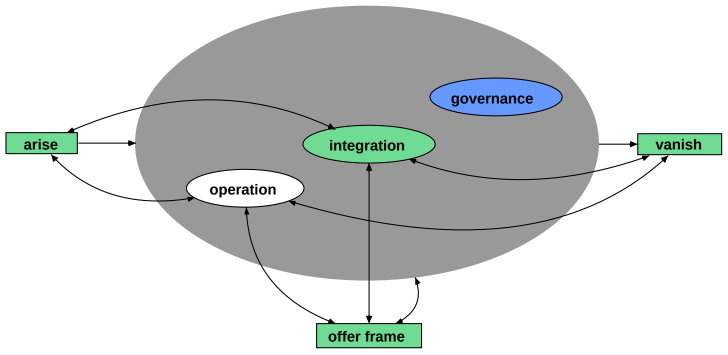

Fig. 3. Open System Unit: Integration Processes.

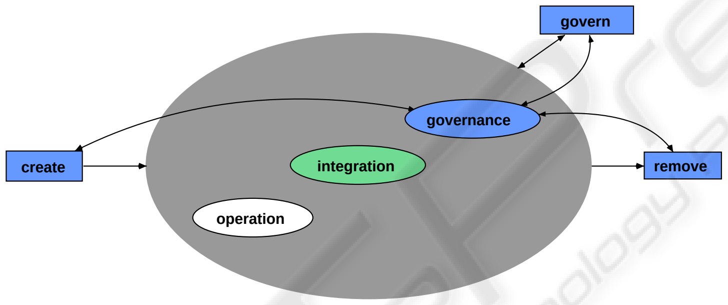

Fig. 4. Open System Unit: Governance Processes.

The modelling of complex systems requires different levels of abstraction. The contents at each level should be described in a way that offers a largely complete and homogenous picture of the selected perspective for this level of abstraction. From this premise we derive our architectural proposal for multi-organization systems.

3.1 Overview

In focussing on the organization as the core unit of the architecture, three necessary levels of examination directly follow: The organization itself, its internals and its environment. The architecture shall include multiple organizations and each of these may have different (conceptual) environments. It follows the necessity of a fourth architectural level as a system closure for the integration of all environments.

Interestingly, this identification of four levels resulting from rather technical considerations is confirmed by organization theory according to Scott [18]. The internals of an organization correspond to the *socio-psychological level* of organization theoretical analysis where the behaviour and relations between individual members of the organization is examined. From this perspective, organizations are regarded as contexts. The organization as a discrete entity of its own appears at the *organization structure level*

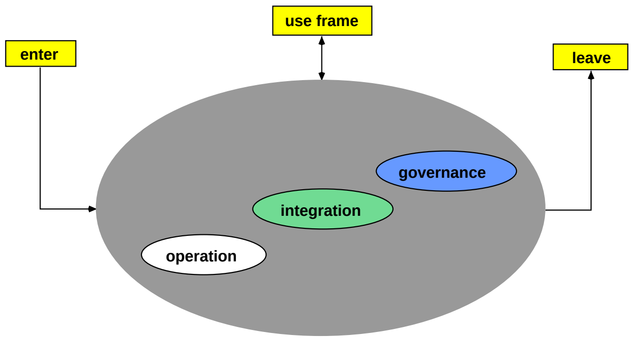

Fig. 5. Open System Unit: Peripheral Processes.

where the structural properties and social processes that characterize an organization and its subdivisions are studied. The *ecological level* finally focuses on characteristics and actions of organizations as corporate actors that operate in even more global networks of relations. For the ecological level, further refinements are possible. Among these the concept of *organizational field* is the most comprehensive one referring to immediate environments of organizations while the *society* offers a common frame for organizational fields.[4]

As a consequence, our reference architecture for multi-organization systems consists of four levels of system units. In order to emphasize that these units are particular instances of the universal scheme of system units according to Section 2, they are termed *organizational units*. Figure 6 shows an overview of the architecture as nested organizational units.[5]

Each department is exclusively assigned to an organization. Each organization consists of multiple departments and operates on multiple organizational fields. Each field hosts multiple organizations and is embedded in the single integrative society.

Departments as the lowest level units of abstraction embody the connection to multi-agent system technology. Each department is a multi-agent system. This perspective only makes sense if a department fulfils an organizational purpose. As described in the introduction of this paper, a considerable part of multi-agent system research of the past years was devoted exactly to this aspect. Thus, we arrive at a seamless transition from agent- to organization-orientation. *All* organizational units of the reference architecture can be regarded as logical units (nevertheless embodied by explicit software constructs, e.g. being agentified) that are built upon physical multi-agent systems.

[4] This distinction according to Scott is a refinement of the prevailing distinction in *micro*- and *macro-level* where the first corresponds to the social-psychological level and the latter encompasses all other levels.

[5] The model has a reference net semantics. Reference nets show some extensions compared to conventional coloured Petri nets [14]. They implement the nets-within-nets paradigm where a surrounding net (the system net) can have *nets as tokens* (the object nets). Reference semantics is applied, so these tokens are *references* to *net instances*. *Synchronous channels* allow for communication between net instances.

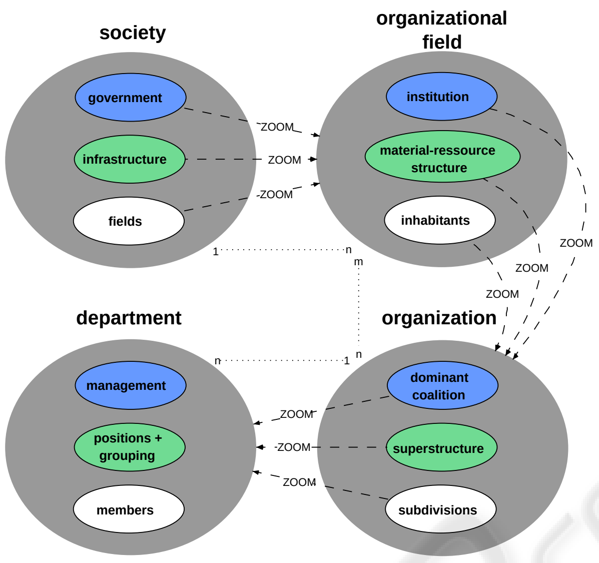

Fig. 6. Multi-Organization Software Architecture: Overview.

3.2 Architectural Levels

Due to space limitations, no complete description of the architectural levels is possible in this paper. Instead, we present a short summary of the characteristics of each level in order to sketch the conceptual differences.

– **Society.** The society as the highest architectural level embodies the closure of the system and thus has no super-ordinate units. It embeds organizational fields. These are connected through a field infrastructure for interaction and migration between fields. The government is a legal authority that holds and enforces system-wide (field-spanning) societal laws.
– **Organizational Field.** DiMaggio characterizes organizational fields as critical units bridging the organization and society level in the study of social and community change [7]. For example, state regulations directed at individual organizations are typically mediated by field level structures such as trade associations.
Concerning the characteristics of an organizational field, two broad categories are distinguished. The material-resource structure characterizes the field as a stock of resources and source of information, which provide the primary premise under which organizations as inhabitants of the field come together. However, material-resource environments always rest on an institution. As Scott puts it, institutions are composed of *regulative*, *normative*, and *cultural-cognitive* elements that together with associated activities and resources provide stability to social life [18].
– **Organization.** Organizations put particular emphasis on an organizational structure and an organizational authority. Mintzberg [15] for example identifies five fundamental types of organizational subdivisions (*operating core, middle line, strategic*

apex, technostructure, support staff) that are integrated into an organizational superstructure. It is built up by grouping individual positions into units and units into ever larger units until the hierarchy is complete.
Each organization has an authority that is in charge of power and setting the organizational goals. Cyert and March [6] come up with a quite broad concept. Organizations are viewed as being composed of various and varying coalitions, each of which seeks to impose its preferences onto the larger system. If none of them succeeds, they seek as allies other coalitions whose interests are related. Finally, a conglomerate will arrive at a mutually acceptable agreement and at the same time will be influential enough to constitute the dominant coalition of the organization.

– **Department.** The requirement of departments being exclusively assigned to organizations is a logical one. Departments of different organizations need not be disjoint and might for example acquire their members from the same physical multi-agent system. Nonetheless, it is crucial to distinguish between different departments. This issue has been addressed by multi-agent system technology and corresponding solutions follow the *common organization implementation architecture for open MAS* from [3]. The integrational units as positions and grouping characteristics represent an *organizational layer* that encapsulates organizational specifications. This layer offers *proxies* to which domain agents from an open multi-agent system must connect to act as members in the organization.
Supervision and authoritarian decision making might be woven into the position and grouping specifications or might instead (or additionally) be taken care of by an explicit management.

4 Conclusions

We have presented an extended perspective on current organization-oriented multi-agent system engineering and derived a reference architecture for multi-organization systems. The architecture introduces four types of (logical) organizational units built upon (physical) multi-agent systems.

The rationale for the selection of the particular organizational units of the architecture is deeply rooted in organization theory. The result is a software engineering approach that supports micro as well as macro perspectives and at the same time is accompanied by concepts and constructs that are familiar from real-world social scenarios. In this respect, our proposal is also one step in the direction of supporting a proper *IT alignment* through a homomorphism between real-world and software artefacts.

As a related aspect, we consider our approach to be multi-perspective. One particular software system might appear in multiple instances of multiple types of organizational units at the same time. It all comes down to embedding relations. For example, one software system might relate to a second one like an organization to a field and to a third one like a department to an organization (virtual organization).

Turning to future work, the practical usage of the architecture is the most pressing issue. As a starting point, a Petri net-based model of organizational structures and services is presented in [13]. At the same time it is demonstrated how agent technology can be used as a middleware to deploy the organizational specifications. The modelling approach is general enough to be adapted for arbitrary levels of abstraction and allows to define collective entities and nest them inside each other. Thus it supports the development of multi-level architectures.

References

1. S. Beer. *Brain of the Firm*. John Wiley & Sons, second edition, 1994.
2. L. v. Bertalanffy. General system theory. In *General Systems: Yearbook of the Society for the Advancement of General Systems Theory*, volume 1, pages 1–10. Ann Arbor, MI: The Society, 1956.
3. O. Boissier, J. Hübner, and J. S. Sichman. 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.
4. K. Boulding. General systems theory: The skeleton of science. *Management Science*, 2:197–208, 1956.
5. W. Buckley. *Sociology and Modern Systems Theory*. Upper Saddle River, NJ: Prentice Hall, 1976.
6. R. Cyert and J. March. *A Behavioral Theory of the Firm*. River, NJ: Prentice Hall, 1963.
7. P. DiMaggio. Structural analysis of organizational fields: A blockmodel approach. In *Research in Organizational Behaviour*, volume 8, pages 355–370. Greenwich, CT: JAI Press, 1986.
8. J. Ferber, O. Gutknecht, and F. Michel. 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.
9. C. Girault and R. Valk. *Petri nets for systems engineering: a guide to modelling, verification and applications*. Springer Verlag, 2003.
10. M. Hannan and G. Carroll. An introduction to organizational ecology. In *Organizations in Industry: Strategy, Structure and Selection*, pages 17–31. New York: Oxford University Press, 1995.
11. N. Jennings. On agent-based software engineering. *Artificial Intelligence*, 177(2):277–296, 2000.
12. A. Koestler. *The Ghost in the Machine*. Henry Regnery Co., 1967.
13. M. Köhler and M. Wester-Ebbinghaus. Petri net-based specification and deployment of organizational models. In *Proceedings of the International Workshop on Petri Nets and Software Engineering (PNSE'07)*, pages 67–81, Siedlce, Poland, June 2007. Akademia Podlaska.
14. O. Kummer. *Referenznetze*. Logos Verlag, Berlin, 2002.
15. H. Mintzberg. *Structure in Fives: Designing Effective Organizations*. Prentice-Hall, 1983.
16. L. Northrop. *Ultra-Large-Scale Systems: The Software Challenge of the Future*. Software Engineering Institute, Carnegie Mellon, 2006.
17. L. Pondy and I. Mitroff. Beyond open system models of organization. In *Research in Organizational Behaviour*, volume 1, pages 3–39. CT: JAI Press, 1979.
18. W. R. Scott. *Organizations: Rational, Natural and Open Systems*. Prentice Hall, 2003.
19. M. Wester-Ebbinghaus, D. Moldt, C. Reese, and K. Markwardt. Towards Organization–Oriented Software Engineering. In *Software Engineering Konferenz 2007 in Hamburg: SE'07 Proceedings*, volume 105 of *LNI*, pages 205–217. GI, 2007.