Structure in Threes: Modelling Organization-Oriented Software Architectures Built Upon Multi-Agent Systems

(Short Paper)

Matthias Wester-Ebbinghaus
University of Hamburg
Vogt-Kölln-Straße 30
22527 Hamburg
wester@informatik.uni-hamburg.de

Daniel Moldt
University of Hamburg
Vogt-Kölln-Straße 30
22527 Hamburg
moldt@informatik.uni-hamburg.de

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.

Categories and Subject Descriptors

D.2 [**Software Engineering**]: Architectures; J.4 [**Social and Behavioral Sciences**]: Sociology; K.6 [**Management of Computing and Information Systems**]: Software Management

General Terms

Software management, organization theory and design

Keywords

Multi-organization systems, multi-agent systems, organizational modelling, organization-oriented software architectures

1. INTRODUCTION

To counter the demands of modern software systems as ever increasing in size and complexity, multi-agent system research has come up with organization-oriented approaches that seek to combine local agent autonomy with the assurance of global system characteristics by imposing "organizational facts" onto the system (cf. [12, 1] for an overview of recent and current work in the field).

Cite as: Structure in Threes: Modelling Organization-Oriented Software Architectures Built Upon MAS (Short Paper), Wester-Ebbinghaus and Moldt, *Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008),* Padgham, Parkes, Müller and Parsons (eds.), May, 12-16., 2008, Estoril, Portugal, pp. 1307-1310..
Copyright © 2008, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

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. The importance of organizations as corporate actors themselves when turning to more global (ecological) levels of analysis [10] has been largely neglected [12].

The aim of this paper 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 [4] 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 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 system units. The introduced universal scheme is utilized to propose concrete organizational units that constitute a reference architecture for multi-organization systems in Section 3. We conclude our results in Section 4.

2. OPEN SYSTEM MODELLING

We address software systems that are composed of various subsystems and adopt the modular view of each system as a *unit*. Few system units are fully self-sufficient and thus need to be open to their environments [9]. We comprehend the environment of each system unit again as a system unit (or multiple system units). Thus we trace the relationship of systems to their environment back to the (not necessarily unique or disjoint) nesting of system units. By recursively applying this understanding, we arrive at a hierarchy of system units, each of which is both an embedding environment and environmentally embedded. This conception is in line with Simon's assertion that hierarchical clustering is a fundamental feature of all complex systems [11].

Figure 1 illustrates our understanding of an open system unit as a structure in threes. [1]In order to study the processes of an open system unit exhaustively we have to consider three orthogonal dimensions with three values respectively.

[1]The model has an underlying coloured Petri net semantics, cf. [5]. However, it contains simplifications (explained in the main text) that would have to be resolved in order to obtain a well-formed Petri net model.

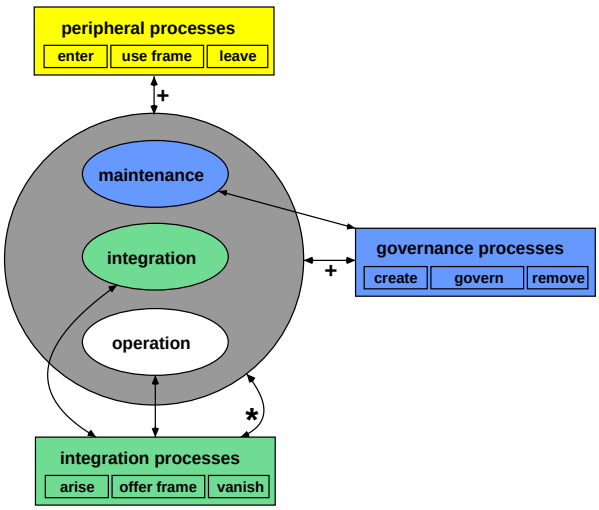

Figure 1: System Unit as Stucture in Threes

- **Directions** System processes impact the system unit from different *conceptual directions*. *Integration processes* impact the system unit bottom-up ("from below"). They include actions of and interactions between the embedded system units of the system unit in focus. *Peripheral processes* impact the system unit top-down ("from above"). They include actions and interactions of the system unit in focus (which in turn have to be mapped onto its internal system units) in relation to its embedding system unit(s). Finally, *governance processes* impact the system unit in focus "at its own level". They fulfil the purpose of manifesting, preserving and enforcing the rules of the system and supervising the evolution of these rules over time.

- **Basic Operations** The evolution of the system unit in focus solely depends on its internal system units. The basic operations that system processes relate to are the *adding* of new internal units, the *removal* of former ones, and the *modification/usage* of existing ones. These basic operations are enlisted for each process class respectively with different names indicating their different contextual function.

- **Internal System Units** The internal system units of the system unit in focus are classified according to the processes in which they are involved. *Operation units* are those that undertake the primary activities and can be considered as the intrinsic parts of the system in focus. However, it is the connections and interdependencies between its internal system units that define the overall behaviour of a system unit. The *integration units* see to it that the singular operation units are integrated into a joint system in the first place. They define the means by which the operation units may participate in the system and so take care of their technical embedding. The strategic embedding is taken care of by *governance units*. They watch over the system unit's rules, laws and regulations concerning structural and behavioural aspects and exploit mechanisms for their enforcement.

  Process participation of internal system units is modelled by arcs. The arcs connecting to the outer circle are shortforms that include all three cases for the inner circles respectively. An arc indexed with a plus indicates that one or more substitutes from the associated class of system units are involved. Analogously, a star indicates that zero or more substitutes are involved.

The nesting of system units is achieved by recursively overlaying the system processes of an embedding system unit with the peripheral processes of embedded system units. [2]

3. REFERENCE ARCHITECTURE

We advance an architectural proposal for multi-organization systems. It is based on the universal scheme of open system units from the former section and introduces particular embodiments according to architectural level.

3.1 Overview

The architecture consists of four levels of *organizational units* (particular embodiments of the universal scheme of open system units) and is illustrated in Figure 2.

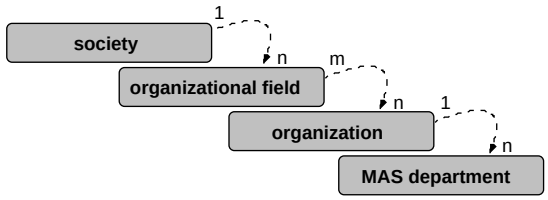

Figure 2: MOS Reference Architecture: Overview

The organizations are the central units of the system with respective sets of exclusively assigned departments as their internals and multiple organizational fields as their environments. The society serves as the system closure and integrates all organizational fields.

These four levels are in line with Scotts classification of analysis levels in organization theory [10]. He identifies the *socio-psychological level*, the *organization structure level*, and the *ecological level* where the latter is further refined into the *organizational field level* and the *societal level*.

3.2 Architectural Levels

No complete description of the architectural levels is possible in this paper. We omit the explicit distinction between the basic operations (add, remove, modify). All arcs that are not indexed with a star are implicitly to be considered as being indexed with a plus. In order to enrich the models with additional semantics, we introduce different arc types for process participation according to Figure 3.

Department. Figure 4 displays a department. From a software engineering perspective the departments are multi-agent systems and close the connection to agent-oriented technology.

[2]Following our specific modelling approach, this overlay can be achieved by turning from conventional coloured Petri nets to *reference nets* [7]. Reference nets 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 synchronous communication between net instances.

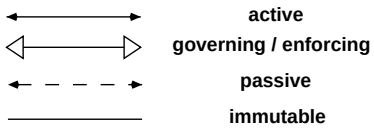

Figure 3: Arc Types for Process Participation

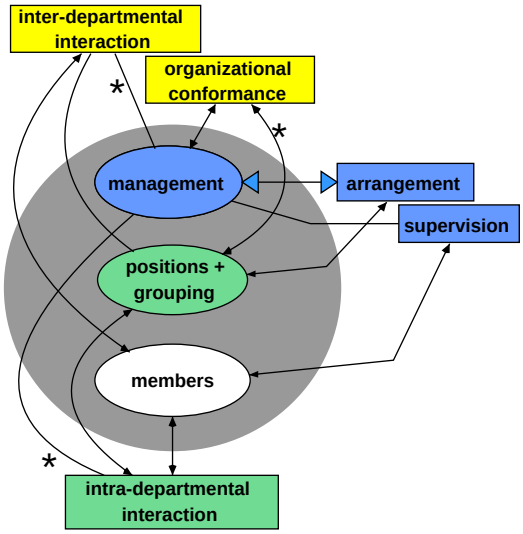

Figure 4: Architectural Level: Department

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 and their affiliations to different organizations. This issue has been addressed by multi-agent system technology and corresponding solutions follow the *common organization implementation architecture for open MAS* from [1]. The integration 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. In this sense, *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.

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. However, it is only the position and grouping constellations that the management may fully govern. The members are black boxes that may only be surveyed, sanctioned or dismissed.

Organization. Figure 5 displays an organization. From a software engineering perspective, the organizational level is the core part of the reference architecture. The level of abstraction of this architectural level determines the scope of the overall system.

Different purposes, requirements and needs of the organization are mapped onto its functional departments. Mintzberg

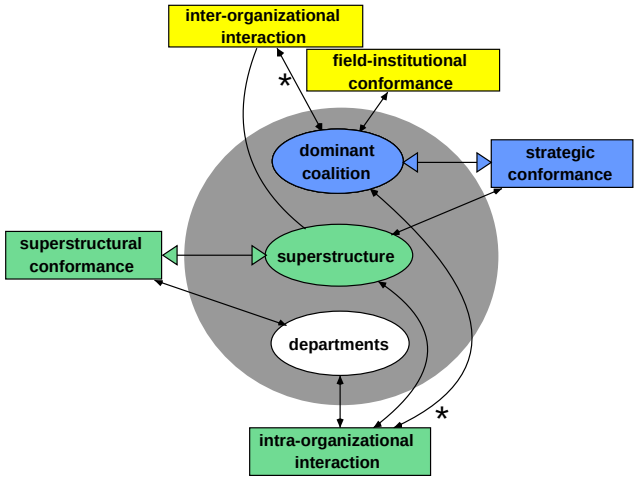

Figure 5: Architectural Level: Organization

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 [8].

Each organization has an authority that is in charge of power and setting the organizational goals and strategies. Cyert and March come up with a quite general concept [2]. 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.

Superstructure as well as all functional departments are white boxes from the perspective of the organization and may be fully governed.

Organizational Field. Figure 6 displays an organizational field. From a software engineering perspective, each field represents a distinct living space for organizations and constitutes a consistent perspective on a part of the overall software system.

An organizational field consists of organizations that together constitute a (legally) recognized area of institutional life: suppliers of core as well as related services and products, consumers, facilities for mediating the exchange of goods and services, and agencies with regulatory oversight [3].

Concerning the characteristics of an organizational field, two broad categories are distinguished according to Scott [10]. The *Material-resource structure* emphasizes the fact that organizations are production systems and as such require resource and energy inputs. The environment is viewed as a stock of resources and a source of information. These provide the primary premise under which organizations come together and give rise to *technical controls*.

However, a material-resource environment always rests on an *institution*. Institutional conceptions of environments emphasize the fact that organizations create and recreate their social reality. Institutions are composed of *regulative*, *normative*, and *cultural-cognitive* elements that together with associated activities and resources provide meaning and stability to social life and give rise to *institutional controls*.

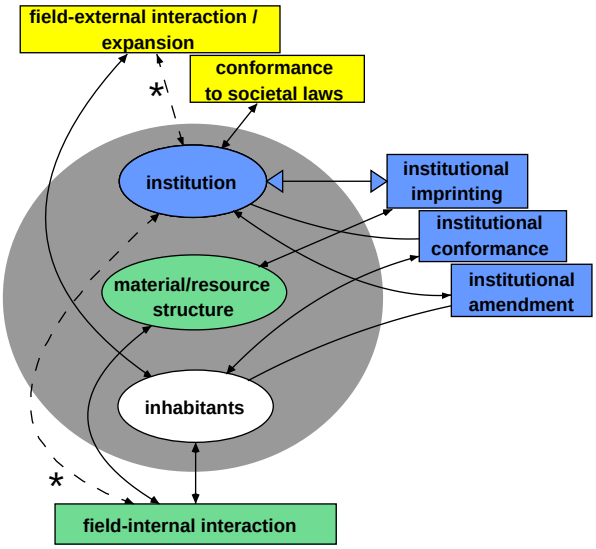

Figure 6: Architectural Level: Organizational Field

The institution governs the material-resource structure as white boxes, channelling the circumstances under which inhabitants may come together. The inhabitants themselves are black boxes, whose habits and practices can only be monitored and potentially rewarded or sanctioned.

Society. Figure 7 displays the societal level as the overarching closure of the software system.

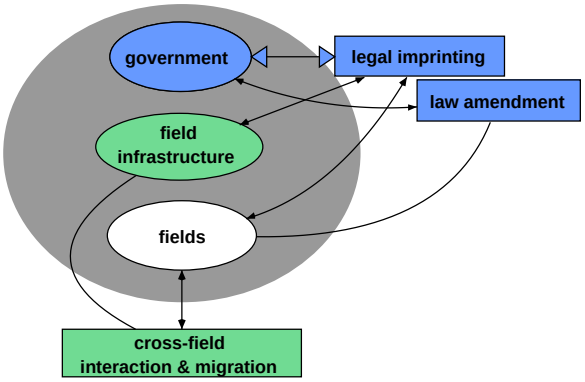

Figure 7: Architectural Level: Society

Organizational fields were characterized as distinct living spaces for organizations. The society contains various fields that allow to represent different (and possibly opposing) requirements and functions of the overall software system. It further allows certain transitions between fields that are both implemented and restricted by the field infrastructure.

A government is responsible for setting and implementing the global, field-spanning laws of the system. Both the field infrastructure and the fields themselves are governed as white boxes.

4. CONCLUSION

We have presented an extended perspective on current organization-oriented multi-agent system engineering and derived a reference architecture for multi-organization 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.

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 [6]. 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. In addition, the approach allows to define collective entities and nest them inside each other and thus supports the development of multi-level architectures based on open system units as presented in this paper.

5. REFERENCES

[1] 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.

[2] R. Cyert and J. March. *A Behavioral Theory of the Firm*. River, NJ: Prentice Hall, 1963.

[3] P. DiMaggio and W. Powell. The iron cage revisited: Institutional isomorphism and collective rationality in organizational fields. *American Sociological Review*, 48:147–160, 1983.

[4] 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.

[5] C. Girault and R. Valk. *Petri nets for systems engineering: a guide to modelling, verification and applications*. Springer Verlag, 2003.

[6] M. Köhler and M. Wester-Ebbinghaus. Closing the gap between organizational models and multi-agent system deployment. In *Multi-Agent Systems and Applications V*, volume 4696 of *Lecture Notes in Artificial Intelligence*, pages 307–309. Springer-Verlag, 2007.

[7] O. Kummer. *Referenznetze*. Logos Verlag, Berlin, 2002.

[8] H. Mintzberg. *Structure in Fives: Designing Effective Organizations*. Prentice-Hall, 1983.

[9] 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.

[10] W. R. Scott. *Organizations: Rational, Natural and Open Systems*. Prentice Hall, 2003.

[11] H. Simon. The architecture of complexity. In *Proceedings of the American Philosophical Society*, volume 106, pages 467–482, 1962.

[12] 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.