

# On the autonomy of software entities and modes of organisation

Alan Colman and Jun Han

Faculty of Information and Communication Technologies  
Swinburne University of Technology  
Melbourne, Victoria, Australia  
{acolman,jhan}@swin.edu.au

---

## Abstract.

As software becomes more complex and needs to operate in more open environments, the relationships between the encapsulated entities that constitute the software can become non-deterministic. In a number of branches of computer science, organisational mechanisms and structures have been seen as a way to coordinate the complex behaviour between software entities. In particular, organisational abstractions have been viewed as a way of handling complexity in various multi-agent system methodologies, and various object-oriented and role modelling methodologies that focus on collaborations. Organisational mechanisms and structures need to be appropriate to the types of software entity being organised. We define five levels of autonomy and discuss the relationship of these levels to different mechanisms and modes of software organisation. This paper proposes that the degree of the autonomy and capability of the entities determines what organisational abstractions are appropriate. In particular, it is posited that organisation structures based on roles with differentiated autonomy are necessary in complex open systems.

---

## 1 Introduction

As software becomes more complex and needs to operate in more open environments, the relationships between the encapsulated entities that constitute the software can become non-deterministic. There may be many reasons for this indeterminism. The software may rely on third-party components or services with uncertain performance; the communication channels between distributed components may be of variable quality; the system might include embedded subsystems with stringent computational resource constraints; the components may be coupled to an uncertain physical world; a mixed-initiative system may have unreliable or variable humans in the loop; and so on.

In a number of branches of computer science, organisational mechanisms and structures have been seen as a way to coordinate the complex behaviour between software entities. In particular, organisational abstractions have been viewed as a way of handling complexity in various multi-agent system methodologies, and various object-oriented and role modelling methodologies that focus on collaborations. Organisational mechanisms and structures need to be appropriate to the types of software entity being organised. This paper proposes that the degree of

the autonomy and capability of the entities determines what organisational abstractions are appropriate. We define five levels of autonomy and discuss the relationship of these levels to different modes of software organisation.

The structure of this paper is as follows. In Section 2 we define what we mean by *organisation* and discuss why an explicit notion of this concept is important for open, complex software systems. Section 3 defines five levels of *autonomy* and discusses the relationship between the autonomy and capability of the software entities, and the amount of *structure* needed in the organisation. Section 4 introduces a classification scheme for *mechanisms* of organisation in terms of the directness of the mechanism; and whether the mechanism focuses on individual entities or is globally applied. Section 5 characterises some major *modes* of organisation in terms of the concepts discussed in sections 3 and 4; that is in terms of the levels of autonomy/capability of the entities and the mechanisms of organisation. Section 6 discusses the need for complex organisations to be built from software entities with differentiated autonomy and capability. Section 7 briefly discusses related work and Section 8 concludes.

## 2 What is organisation?

A description of a system's organisation is a description of the relationships between elements in that system. Organisation is an abstraction over associations. There are many definitions of organisation. Parunak and Brueckner [25] suggest three complementary aspects of organisation based on information entropy, process, and (emergent) structure. Organisation<sub>1</sub> ( $O_1$ ) is the inverse of the amount of entropy in the system as measured by some observed regularity (e.g. spatial, functional or temporal);  $O_2$  is the *processes* by which  $O_1$  increases in time; and  $O_3$  is the *structure* resulting from  $O_2$  which can be measured with  $O_1$ .

This definition nicely binds the physical and informational aspects of organisation together but it has two shortcomings. Firstly the measure of organisation ( $O_1$ ) is itself a state-based metric. A more system-theoretic definition of organisation can be found in Maturana and Varela [19]. In their definition, organisation is the set of relationships that maintain the viability of a complex biological system in a changing environment. The *viability* of systems is the ability to adapt and survive in a changing environment both ontogenically (restructuring of the individual system or unity) and phylogenetically (evolution of the species or system type through the reproduction of variants). In complex systems these associations can be seen as *defining* the system. For example, in a complex multi-cellular system such as an animal, cells are continually dying and being replaced. What stays constant in such biological systems is the relationships between the roles played by these cells. It is these relationships that define a system as a unity, and determine the dynamics of interaction and transformations (the ontogenic adaptation) which unity may undergo [20]. Organisation is what maintains the system as a *viable* entity in a changing environment.

The second limitation of the definition in [25] is that it only addresses *emergent* structures such as those in natural systems. For software systems that are *designed* to achieve goals the above definition needs to be modified to take account of the organisation's purpose. Designed organisations also defined in terms of the division of tasks [22]. The process ( $O_2$ ) can include activities to deliberately modify the structure ( $O_3$ ) to achieve the system's purpose. In designed systems organisational descriptions are means-end functional descriptions and are at a higher level of

abstraction than either state or process perspectives. For example, in object-oriented design these perspectives are captured, respectively, in class and collaboration diagrams. Neither of these representations captures the *purpose* of the entities represented. The concept of a role, on the other hand, is a division of task that *does* capture the purpose of the entity. A role is an interface of an object that satisfies responsibilities to the system as a whole. Roles are the nodes of designed organisational structures.

The shortcoming of many software descriptions is that they reduce the description of organisation to just the topological structure or to just the process. The perspective of process and state are partial perspectives. A complete organisational description would need to indicate how goals are *transmitted* through the system; how the entities are *coordinated* to avoid performing extraneous or mutual destructive activity; how the system *changes* in response to changing goals and environmental perturbations; and how the system maintains its organisational viability. To be useful as an input to the design process, such descriptions need to be based on principles of organisation that are more than descriptions of particular interactions in a domain specific system. Certain architectural styles [29] and system patterns [3] are examples of the expression of such principles in domain independent form.

In summary, we define organisation as the relationship of roles in the system, and the processes that maintain the viability of these relationships in response to changing goals and changing environments. The mode of organisation of a software system will depend on the nature of the entities that play the roles in that system. In the next section we define two general characteristics of software entities that determine the appropriate mode of organisation.

### 3 Autonomy and capability in software entities

**Autonomy.** In his comparison of the natural and artificial worlds Simon [32] sees quasi-autonomy from the outer environment as an essential characteristic of complex systems. Complex systems are aggregations of “stable intermediate forms”. The quasi-autonomous system or subsystem maintains a homeostatic relationship with its environment (including other subsystems). Autonomous entities/systems however always operate within a set of environmental constraints that restrict the entity’s autonomy. Organisational mechanisms and structures can be viewed as the beneficial restriction of an entity’s autonomy in order to maintain viability and achieve higher/social level goals.

We can define five ‘levels’ of autonomy based on the constraints to which an entity is subject. These levels are not strictly progressive. For example, a badly behaved object may exhibit constraint autonomy but not have intentional autonomy. The levels are:

1. **No autonomy.** The entity is told what actions to execute and it always attempts to execute them. For example, if the entity was responsible for making a cup of tea, then each of the actions to achieve that goal would be prescribed.
2. **Process autonomy.** The entity is given a task to perform in the form of a goal state but it has some autonomy in what steps are executed in order to achieve that task. In software terms, an object could be regarded as autonomous in this sense as its implementation is hidden behind its interface. In our tea-making entity example, the entity could make the tea using a tea-bag or a pot.

3. **Goal-state autonomy.** The entity is given an external goal, which may be satisfied by a number of states. For example, our entity is given the goal of satisfying the thirst of an external entity. In response, our entity could make a cup of tea or make a cup of coffee. A software example of goal-state autonomy would be an operating system that maintains processing capacity by deciding the run-time priority of processes.
4. **Intentional autonomy.** The entity has the freedom to decide whether or not to satisfy external goals — it has (or we ascribe to it) its own intentions. In our tea-making example, it may or may not provide a drink. A *cooperative* entity will fulfil the request if it can. A *competitive* entity only does so if it receives sufficient reward. In the software domain, proactive software agents might exhibit intentional autonomy. Cooperative agents in a *cooperative distributed problem solving* (CDPS) attempt to collaborate to system level goals, whereas competitive agents in a market-based system attempt to maximise the own utility.
5. **Constraint autonomy.** An entity that exhibits constraint autonomy is prepared to violate norms or even rules to achieve its goals. A greedy software agent may take no account of the computational resource it consumes. A malicious agent may deliberately try to harm other agents or the system environment itself.

The table below summarises these levels of autonomy, and the types of software entity that exhibit such autonomy.

**Table 1. Levels of autonomy in various types of software entity**

Level of autonomy \ Entity type	Process	Goal state	Intentional	Constraint
Unencapsulated algorithm	×	×	×	×
Encapsulated object	✓	×	×	×
Intelligent agent/object	✓	✓	×	×
Proactive cooperative or competitive agent	✓	✓	✓	×
Anti-social or malicious agent	✓	✓	✓	✓

**Capability.** The *capability* of a software entity can be viewed as the effective exercise of *autonomy* at a particular level. The more autonomy an entity has, the more capability it will need in order to maintain a stable relationship with its environment, and consequently achieve its goals. Wooldridge [34] defines three levels of flexible autonomous action within an environment. Firstly, a *reactive* system maintains an ongoing interaction with its environment, and responds to changes that occur in it. Secondly, *proactive* systems exhibit goal-directed behaviour. Entities (agents) in proactive systems recognise opportunities and take initiative to set and achieve goals. Thirdly, where agents are operating in a multi-agent environment, agents may need to interact with one another (and possibly humans) in order to achieve their goals. This *social* ability requires inter-agent communication. This communication can occur via a language or by agents changing the mutual environment. We will not discuss the concept of *capability* further here, other than to note that the concepts of *role* and *capability* can be separated [11]. Within limits, objects/agents of varying capability can play the same role in an organisation. For example, in a human bureaucracy, various roles in the organisation are played by employees but these employees will vary in their ability to perform their assigned roles.

**Structure.** Many accounts of complex systems such as Simon's [2] concept of 'hierarchy' [32], Cybernetics and General Systems Theory [6,33] posit that viable complex systems are organised as hierarchies of semi-autonomous entities. As (natural) systems evolve, successive layers of complexity and control are added<sup>1</sup>. While many analogies for organisation in software have arisen from natural systems, [31] and [6] have derived similar concepts from management theory and operational research into human organisations [4,6]. Agre [1] notes that Simon (1957) outlined many ways in which social organisations compensate for the "limited rationality" of their members. "The orchestration of numerous workers within a larger organization, Simon argued, compensates for the individual's limited capacity for work. Likewise, the division of labor and the assignment of specialized tasks to individuals compensates for their limited abilities to learn new tasks. The flow of structured information through the organization compensates for their limited knowledge, and the precise formats of that organization, together with the precise definition of individual tasks, compensate for individuals' limited abilities to absorb information and apply it usefully in making decisions. Finally, Simon believed that the hierarchical structure of bureaucracies compensates for individuals' limited abilities to adopt their own values and goals."

There tends, therefore, to be an inversely proportional relationship between the autonomy/capabilities of the entities that play the roles in the organisation, and the amount of structure needed. Organisational hierarchy can compensate for the lack of autonomy and capability in software entities — but at the cost of some flexibility and adaptability. The less constrained the environment is in which the entity operates, the more capability that entity needs to have in order to maintain its homeostatic relationship with its environment.

Returning to our schema for types of software entities that exhibit various levels of autonomy and capability, we can now say something about the type of organisational structures required for such entities. In the following sections we will characterise *mechanisms* and *modes* of organisation. These *mechanisms* are the intentional and unintentional processes that create the organisational structure. *Modes* are prototypical patterns of organisation appropriate to entities of varying autonomy and capability.

## 4 Mechanisms of Organisation

As we noted in Section 2, an organisational description needs to take account of the process ( $O_2$ ) by which organisational structure ( $O_3$ ) is created. In natural systems

---

<sup>1</sup> Heylighen & Joslyn [15] nicely summarise this hierarchy of systems from a cybernetic perspective: "A control loop will reduce the variety of perturbations, but it will in general not be able to eliminate all variation. Adding a control loop on top of the original loop may eliminate the residual variety, but if that is not sufficient, another hierarchical level may be needed. The required number of levels therefore depends on the regulatory ability of the individual control loops: the weaker that ability, the more hierarchy is needed. This is Aulin's 'law of requisite hierarchy'. . . when the variety becomes really too great for one regulator, a higher control level must appear to allow further progress. We call this process a metasystem transition, and propose it as a basic unit, or "quantum", of the evolution of cybernetic systems. It is responsible for the increasing functional complexity which characterizes such fundamental developments as the origins of life, multicellular organisms, the nervous system, learning, and human culture."

this structure is purely emergent, but in artificially designed systems organisational structure can be intentionally created. The structure can be created using different mechanisms either at design-time or run-time. These mechanisms can be characterised according to the two dimensions: scope and type of control. The *scope* of these mechanisms ranges from the creation of explicit *control-structures* that constrain the *individual* entities playing roles in the organisation, through to the system-wide or subsystem-wide *protocols* that define *global* rules between the entities. Organisational mechanisms (for example defining bureaucratic relationships between roles) can also be characterised according to control-type as direct or indirect. Direct mechanisms define what an entity may/should/must do. Indirect mechanisms constrain the entity's actions through the allocation of resources or the definition of norms/policies/rules etc. Table 2 below gives examples of various mechanisms of organisation classified according to these two dimensions.

**Table 2. Examples of organisational mechanisms classified by control type and scope of control**

Scope \ Control type	Structure that constrains individual component playing role	System-wide protocol
Direct control	Command / goal setting / role descriptions	Prescriptive rules
Indirect control	Resource allocation Individual policies	Group policies Norms

In terms of intentional agent organisations (societies) Castelfranchi [8] points out that organisation based on roles is only one method of social control. These mechanisms include normative control based on obligations, sanctions and norms; and economic control based on incentives and resource scarcity. Coordination is achieved through “mutual adjustment” [21] rather than rigid structure. However, achieving first-order goals or system-level outcomes solely through the design of social norms and laws can be a computationally hard task even in simple domains [30]. Wooldridge [34] points to the difficulty of designing norms and laws in advance because we cannot know all the characteristics of the systems at design time; the goals of agents change; and as the complexity of the system increases design laws to cope with the number of ‘trajectories’ becomes increasingly difficult. Wooldridge suggests the use of emergent norms based on ‘successful’ strategies of agents and techniques for determining these desirable strategies. However, it is difficult to see how such emergent organisation can be made compatible with first-order system goals. Some degree of explicit organisational structure in the form of a role structure is probably necessary to achieve this.

In many complex systems, a combination of these mechanisms of organisation will be necessary. It can be postulated that open software organisations (as with human organisations [21]) that use indirect control and mutual adjustment are more adaptable (and thus robust) but less formally goal-oriented than closed organisations that use direct-control [25]. Designing a system that is both goal-oriented *and* adaptive involves finding organisational mechanisms that achieve an optimal balance on these dimensions.

Figure 1 below shows the relationship between types of actions, the constraints on the autonomy of action, and mechanisms for those constraints. The organisational mechanisms (boxes in bold) define the space of possible actions for an entity. The degree of autonomy of the entity determines how constrained the entity is by those

mechanisms. For example, a goal-oriented organisation made up of entities with only process- autonomy would be constrained to the inner-most space. On the other hand, an organisation of benevolent but self-interested entities would have a wider scope of action. As can be seen from the diagram, direct and indirect control mechanisms are complementary rather than alternate mechanisms for defining a space of desirable actions.

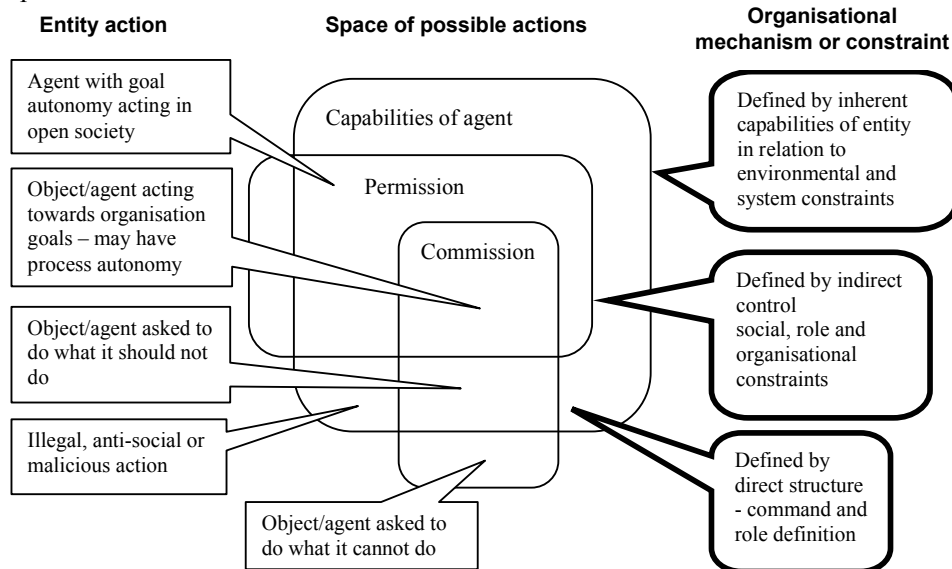


Figure 1. Constraint of autonomy defined by organisational mechanisms

## 5 Modes of Organisation

In Section 3 we pointed out the inversely proportional relationship between the autonomy/capability of software entities, and the degree of structure needed. In Section 4 we listed some of the direct and indirect mechanisms by which an organisational structure can be created. In this section we will characterise some modes (general patterns) of organisation in terms entity type and mechanism of organisation.

As we point out in Section 3 above, the degree of *autonomy* exhibited by components can vary from objects that respond deterministically to requests, to components or agents with varying degrees of autonomy — autonomy to take action, to set goals and even to violate certain types of constraint. The *capability* of software components can range from passive objects that respond deterministically to a set range of inputs, to intelligent agents that actively sense their environment and deliberate on courses of action. In this schema (contrary to the view in [35]) there is no sharp distinction between objects/components and agents. *Both object and agents can play roles* as we have defined them in Section 2. The type of component combined with the different types of control or constraint (organisational mechanisms) that restrict component autonomy result in different modes of organisation. We can characterise such modes by their predominant method of organisational structuring. In order of increasing autonomy of the components/agents, examples of such modes of organisation are:

- Command based organisation - an organisation formed as a command hierarchy would have strong central goals while the components have little autonomy.

- Fixed role-based organisation - an organisation based on roles would maintain central goals, but components or agents would have autonomy of action to carry out their roles.
- Adaptive role-based organisation - an organisation where there is a loose coupling between components and roles. Roles may be performed by different components. Components may have the autonomy to adopt roles. Relationships between roles, and thus components, can also change.
- Emergent cooperative organisation – a protocol-based organisation gives the agent goal-autonomy and process-autonomy within certain constraints (norms, laws, access to resources etc). Agents share social goals and act cooperatively within the constraints to achieve these goals.
- Competitive - there may be no explicit social first-order goals or organisation. In a competitive organisation framework, entities adopt their own goals but also have an interest in maintaining the viability of the system environment. They therefore follow rules and norms. These constraints would be aimed at maintaining the viability of the system and at the optimisation of utility.
- Laissez-faire - agents would be able to act and set goals as they please, but also violate norms and rules if they perceive such action to be in their interest. Structure would still emerge but be based on individual agent’s capability, power and ability to access resources – the ‘law of the jungle’ prevails.

Organisational structure is designed or emerges from interactions between components of varying autonomy. These interactions are mediated by structuring mechanisms. Table 3 below presents the relationship between degrees of component autonomy and modes of organisation.

**Table 3. Autonomy of software components and modes of organisation**

Organisation	Command hierarchy	Fixed roles based organisation	Adaptive roles based organisation	Defined protocols / emergent structure	No universal protocols (laissez-faire)
Autonomy of entity					
None	✓	✓	✓	✓	✓
Process	x	✓	✓	✓	✓
Goal-state	x	x	x	✓	✓
Intentional	x	x	x	✓	✓
Constraint	x	x	x	x	✓

← Strong organisational goals / Designed structure      Weak organisational goals / Emergent structure →

In any real-world organisation/society, a combination of these modes of organisation is likely to be present. For example, an organisation such as an army that might be primarily characterised as a command-and-control organisation, would also be influenced by role-based and emergent structures. The formal authority structure in an army is likely to be modified by informal networks of communication and influence, established norms of behaviour, capability of the officer-agents etc. In designed closed computer systems the command-and-control mode of organisation predominates. However, the achievement of goals in such an organisation is qualified by its ability to access resources and so on. As computing moves to more open architectures, role-based and emergent organisational modes will become more important.

## 6 The importance of differentiated structure

In section 4 we discussed the necessity for a software system to have an organisational structure of roles to compensate for the limited rationality of the system's entities — particularly if that system is to achieve first-order, system-wide goals. We further postulate that for complex systems, such organisation structure also needs to be differentiated in terms of the *capability* and *autonomy* of the entities that comprise the system.

Mintzberg [22] has shown that in human organisations the higher the level a role is in the organisational structure, the less formalised and standardised the behaviour required of that role. At these higher levels, the relationships between the role and other roles in the organisation, and between the role and the external environment, are much more variable than the relationship at lower levels. In cybernetic terms, the *variety* of the higher-level role is greater, and therefore the *capability* of the controller (object/agent/person) playing that role must be greater. If the system is to remain viable, the internal complexity (variety) of the controller must match the external complexity of its environment [5].

These higher level roles also exhibit greater *autonomy*. In Section 4 we characterised more open systems as those where more types of autonomy may be exhibited by the entities in that system. This implies that an open system, such as an agent-based system, can consist of entities with *varying* degrees of autonomy and capability. Indeed, to create a viable organisational structure that can achieve system level goals, different entities *must* have varying degrees of autonomy if the entities are to have capability commensurate with the complexity of their respective environments. *Entities should have enough autonomy but not too much autonomy.* The higher the role played by the entity is up the hierarchy, the more autonomy and capability that entity needs to be able to adapt to environmental perturbations. In human bureaucracies, the directors of the organisation have great autonomy to act with constraints imposed by their role and the environment. Directors have goal-state and intentional autonomy. As we descend the role hierarchy, the autonomy of the entities playing those roles becomes much more constrained. A line manager may only have process autonomy while a production worker may process no autonomy in their role. Examples of software architectural approaches that incorporate entities with differentiated autonomy can be found in [11,12] and [14].

This need for differentiated structure has implications for software design methodologies. There has been a tendency for the practice of methodologies to be mono-cultural, if not xenophobic. The methodology applied is exclusively either structured *or* object-oriented (“everything is an object”) *or* agent-oriented (“only agents can play roles”). For example, Wooldridge & Jennings [35] regard OO and AO methodologies as incompatible because agents have stronger encapsulation, more autonomy, greater granularity, more unpredictable interactions and more complex internal state than objects. However, most of these differences are a matter of *degree* of autonomy or else are hidden through encapsulation from a system viewpoint.

In our view, by separating the function of the role from the capability of the role-player, role structures provide a unifying concept that allows the combination of agent-oriented approaches and object-oriented approaches in the creation of complex organisations in open environments.

## 7 Related work

The need for *organisational* structure in complex software systems has been recognised in both multi-agent software (MAS) and in role-based methodologies. Much discussion on both autonomy and organisation has taken place within MAS literature (e.g. [9]). This discussion has included proposing agents with variable autonomy and capability rather than having such parameters organisational defined [10,18]. Other work has also noted the relationship between these parameters and various modes or forms of organisation [27]. While these forms of organisation focus on more open systems and may differ from those modes characterised here, such work supports the existence of a close dependency between the autonomy/capability of the agent and the form organisation.

MAS organisations are often conceived as consisting of collection of interacting roles. For example the Gaia [35] methodology for agent-oriented analysis and design has roles as a first-class concept in the analysis stage. In a later development of Gaia [36] recognize that that organisational abstractions need to be explicitly modelled: while a number of MAS methodologies “recognize (to varying extents) the idea that a MAS can be conceived in terms of an *organised society* of individuals in which each agent plays specific *roles* and interacts with other agents according to protocols determined by the roles of the involved agents ... we show that an organization is more than simply a collection of roles (as most methodologies assume), and that in order to effectively build a MAS in organizational terms, further organization-oriented abstractions need to be devised and placed in the context of a methodology”. In Gaia, agents (design level concept) implement a set of roles which are implicit in the design. Other approaches explicitly model roles in the design but do not model organisational structure. [17] uses roles as a method for identifying agents (there is a one-to-one relationship) while [23] see roles as separate implementation concept to agents – agents can adopt roles. Not all methods of role identification are top-down allocation based on an organisational structure. In Prometheus [24] roles are conglomerations of functions grouped using an analysis of coupling and cohesion of those functions. Roles tend to be a feature of methodologies related to object-oriented approaches rather than methodologies whose genesis is knowledge representation [16].

Organizational level abstractions are also *partially* addressed in work on software architecture [3] and frameworks [26,29]. According to Garlan [13], architecture should express “separation of concerns about the functionality of a component from the ways in which a component is connected to (interacts with) other components”. In this sense, as well as representing topology, architectural styles are *abstract* expressions of the management responsibilities and the run-time control of components.

Organisational abstractions are also apparent in control-theoretic architectures [28] in that they separate control from functional processes. Such systems are designed to maintain system viability during anticipated environmental perturbation. Recent work on intelligent control [14] use differentiated capability by adding an adaptive, high-variety loop on top of the operational control loop. These loops, however, are contained *within* each “viable component”. While this allows for the possibility of controllers with differing capabilities within each component, in such control-theoretic approaches the organisational structure of the system is explicitly defined as there is no concept of role.

Our work on the ROAD methodology [11,12] has extended object-oriented role-based approaches [7] to include organisation abstractions. By separating the role

from the object or agent that plays that role, and by creating explicit management (organisational) structures, ROAD is aimed at making systems more adaptable and adaptive. Our approach contrasts with the most MAS approaches to organisational abstraction in that ROAD involves a light relaxation of constraints imposed by pre-existing organisational structures, rather than attempting to create organisational structure from scratch through team and coalition building as in MAS.

## 8 Conclusion

If system level goals are to be achieved and maintained, some form of organisational structure is necessary. Organisation is defined here as the relationship between *roles* in the system, and the processes that maintain the viability of these relationships in response to changing goals and changing environments. Given that highly capable and adaptive humans (compared with software entities) need to form themselves into organisations to achieve complex goals, it would seem likely that software entities will need to be similarly organised (even if they are highly capable and autonomous software agents). The capability/autonomy of the entities playing roles in an organisation is inversely proportional to the amount of structure needed in the organisation. Organisational structures in complex systems should be based on the roles played by entities with *differentiated* autonomy and capability.

## References

- [1] Agre, P. E. "Computational research on interaction and agency - Introduction" in *Computational theories of interaction and agency*, eds. Agre, P. and Rosenschein, S. J. MIT Press, 1995, pp. 1-52.
- [2] Agre, P. E., "Hierarchy and history in Simon's 'Architecture of Complexity'" *Journal of the Learning Sciences*, vol.12(3) , 2003, pp. 413-426.
- [3] Bass, L., Clements, P., and Kazman, R. *Software architecture in practice*, Reading, Mass: Addison-Wesley, 1998.
- [4] Beer, S. *The heart of enterprise*, Chichester Eng., New York : Wiley, 1979.
- [5] Beer, S., "The Viable System Model: Its Provenance, Development, Methodology and Pathology." *Journal of the Operational Research Society*, vol.35(1) , 1984, pp. 7-25.
- [6] Beer, S. *Diagnosing the system for organizations*, Chichester West Sussex, New York: Wiley, 1985.
- [7] Bäumer, D., Riehle, D., Siberski, W., and Wulf, M. "Role Object" in *Pattern languages of program design 4*, eds. Harrison, N., Foote, B., and Rohnert, H. Addison-Wesley, 2000, pp. 15-32.
- [8] Castelfranchi, C. "Engineering Social Order" in *Engineering Societies in the Agents World*, eds. Omicini, A., Tolksdorf, R., and Zambonelli, F. Berlin: Springer, 2000, 1-18.
- [9] Castelfranchi, C., Falcone, R., and Hexmoor, H. *Agent autonomy*, Boston : Kluwer Academic Publishers, 2003.
- [10] Cohen, R. and Fleming, M. "Adjusting the autonomy in mixed-initiative systems by reasoning about interaction" in *Agent Autonomy*, ed. H. Hexmoor, C. C. a. R. F. Kluwer, 2003 , pp. 9-22.
- [11] Colman, A. and Han, J., "Coordination Systems in Role-based Adaptive Software," *Proceedings of Coordination 2005 (Coord'05)*, Namur, Belgium, 2005.
- [12] Colman, A. and Han, J., "Operational management contracts for adaptive software organisation," *Proceedings of the Australian Software Engineering Conference (ASWEC 2005)*, Brisbane, Australia, 2005.
- [13] Garlan D., "Software architecture: A roadmap." *Proceedings of the International*

- Conference on Software Engineering: On the Future of Software Engineering, Limerick, Ireland, 2000*; 2000, pp. 91-101.
- [14] Herring, C. E., Viable software: The Intelligent control paradigm for adaptable and adaptive architecture, PhD Thesis. University of Queensland.2002.
- [15] Heylighen, F. and Joslyn, C. "Cybernetics and Second-order Cybernetics" in *Encyclopedia of physical science and technology*, Anonymous2001
- [16] Iglesias, C. A., Garijo, M., and Gonzalez, J. C. "A survey of agent-oriented methodologies" in *Intelligent Agents - Proceedings of the Fifth International Workshop on Agent Theories, Architectures, and Languages (ATAL-98), Lecture notes in artificial intelligence*, eds. Müller, J. P., Singh, M. P., and Rao, A. S. Heidelberg: Springer-Verlag, 1999
- [17] Kinny, D. and Georgeff, M., "A methodology and modelling technique for systems of BDI agents" *Workshop on modelling Autonomous Agents in a multi-agent world. Springer-Verlag*, 1996, pp. 56-71.
- [18] M. Luck, M., Munroe, S., and d'Inverno, M. "Autonomy: Variable and Generative" in *Agent Autonomy*, ed. H. Hexmoor, C. C. a. R. F. Kluwer, 2003 , pp. 9-22.
- [19] Maturana, H. R. and Varela, F. J. *Autopoiesis and cognition the realization of the living*, Dordrecht, Holland, Boston: D. Reidel Pub. Co, 1980.
- [20] Maturana, H. R. and Varela, F. J. *The tree of knowledge the biological roots of human understanding*, Boston: New Science Library. Random House, 1987.
- [21] Mintzberg, H. *The structuring of organizations: a synthesis of the research*, Englewood Cliffs, N.J: Prentice-Hall, 1979.
- [22] Mintzberg, H. *Structure in fives: designing effective organizations*, Englewood-Cliffs, New Jersey: Prentice Hall, 1983.
- [23] Odell, J., Parunak, H. V. D., Brueckner, S., and Sauter, J., "Changing Roles: Dynamic Role Assignment" *Journal of Object Technology, ETH Zurich*, vol.2(5) , 2003,. 77-86.
- [24] Padgham, L. and Winikoff, M., " Prometheus: A Methodology for Developing Intelligent Agents," *Proceedings of the Third International Workshop on Agent-Oriented Software Engineering at AAMAS 2002*, Bologna, Italy, 2002.
- [25] Parunak, V. and Brueckner, S., "Engineering Self-Organising Applications" *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent System (AAMAS'03)*, 2003, pp. 1-54.
- [26] Riehle, D., Framework Design - A Role Modeling Approach. Dissertation, Swiss Federal Institute of Technology, Zurich. <http://www.riehle.org/computer-science/research/dissertation/>, 2000.
- [27] Schillo, M., Fischer, K., and Siekmann, J., "The Link between Autonomy and Organisation in Multiagent Systems," *Proceedings of the First International Conference on Applications of Holonic and Multiagent Systems (HoloMAS'03). Lecture Notes in Artificial Intelligence*, vol. 2744, 2003.
- [28] Shaw, M., "Beyond objects: A software design paradigm based on process control." *ACM Software Engineering Notes*, vol.20(1) , 1995, pp. 27-39.
- [29] Shaw, M. and Garlan, D. *Software architecture perspectives on an emerging discipline*, Upper Saddle River, N.J: Prentice Hall, 1996.
- [30] Shoham, Y. and Tennenholtz, M., "On social laws for artificial agent societies: off-line design" *Artificial intelligence*, vol.73(1-2) , 1995, pp. 231-235.
- [31] Simon, H. A. *Administrative behavior : a study of decision making process in administrative organization*, New York : Macmillan, 1957.
- [32] Simon, H. A. *The sciences of the artificial*, Cambridge: M.I.T. Press, 1969.
- [33] Skyttner, L. *General systems theory ideas & applications*, Singapore, River Edge, N.J: World Scientific, 2001.
- [34] Wooldridge, M. J. *An introduction to multiagent systems*, New York: J. Wiley, 2002.
- [35] Wooldridge, M. J. and Jennings, N. R., "The Gaia Methodology for Agent-Oriented Analysis and Design" *Autonomous Agents and Multi-Agent Systems*, vol.3(May - June) , 2000, pp. 285-315.
- [36] Zambonelli, F., Jennings, N. R., and Wooldridge, M., "Developing multiagent systems: The Gaia methodology" *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol.12(3) , 2003, pp. 317-370.