Regulating Organizations: The ALIVE Approach*

Huib Aldewereld, Loris Penserini, Frank Dignum, and Virginia Dignum

Institute of Information and Computing Sciences
Universiteit Utrecht
P.O.Box 80089, 3508 TB Utrecht
The Netherlands

Abstract. Regulating organizations requires a fine balance between central control and (local) adaptability. In this paper we report on our approach using explicit organization and coordination models based on the research performed within the European FP7 project ALIVE. One of the principal aims of ALIVE is to combine coordination and organization mechanisms in order to provide a flexible, high-level means to model the structure of interactions between services in the environment. Our main focus is on the implementation and integration of organizational structures and on the translation of (abstract) norms (e.g., laws and regulations) into (concrete) software systems. Such a way of abstracting from low level system complexity by the use of human- and social- oriented system requirements is very promising to cope with requirements changes, e.g., useful to develop adaptive (service-based) systems.

Key words: Organizations, implementation, norms

1 Introduction

The deployment of regulations in human societies and in information systems show remarkable resemblances. Both fields need to cope with relating the abstract level of the regulations with the concrete practice (either the "work floor" of the human organization or the "low-level" software implementation, e.g., using service-based systems). A common tendency when relating the regulations to the practice is to directly connect the top (abstract) level with the concrete implementation, but in this paper we argue that the use of intermediate level(s) allows for a greater flexibility and, at the same time, an increased robustness of the system. This new source of (human- and social- oriented) system requirements pave the way for new challenges in software engineering. That is, recent software engineering approaches have dealt with how to endow single (agent-based) systems with the ability to cope with context changes, without taking

* This work has been performed in the framework of the FP7 project ALIVE IST-215890, which is funded by the European Community. The author(s) would like to acknowledge the contributions of his (their) colleagues from ALIVE Consortium (http://www.ist-alive.eu)

into account that such adaptivity properties can be easier and better studied and handled at organizational level, as often it happens in real life. A challenging aim of this paper is to bridge adaptivity at organizational level.

The research presented in this paper is part of the European FP7 project ALIVE, which aims to create a framework for software and service engineering, based on combinations of coordination and organization mechanisms [1, 3, 13, 14] (providing a flexible, high-level means to model the structure of interactions between services in the environment) and Model Driven Design (providing for automated transformations from models into multiple platforms). Although the main goal of the ALIVE project is to enhance the development and deployment of service-based (information) systems, we will show that the same approach can be applied to the deployment of regulations in human organizations.

The approach taken in the ALIVE project is to gradually translate the regulations from the abstract level of organizational regulations into a system description at the concrete level of service-based implementations. First, the abstract regulations are translated into operational norms and structures, which are more concrete than the regulations themselves. This translation is done by adding operational information to the regulations (i.e., how a given regulation can be achieved in a given context/domain). These operational artifacts [10], however, still abstract from the specific choices needed for the implementation (e.g., different checks to be made, specific system calls, etc.). The use of such an intermediate level is advantageous, because it, in essence, specifies the global objective of the organization in concrete terms, while still describing a family of implementations (i.e., the intermediate level allows for a flexible implementation). This means that this intermediate level contains enough information to make implementational changes without having to go back to the most abstract level of the organizational regulations (i.e., you change the implementation by choosing a different member of the family of implementations that is specified at the operational level).

One way of deploying regulations in an organization is by regimenting the participants of the organization and constrain them in such manners that they can only perform behaviour which the organization considers legal. That means, all possible actions are *a priori* defined by the organization. While, at first glance, this appears to be a fruitful approach, it has the major disadvantage that the system loses much of its flexibility and robustness. If, however, the participants are allowed to perform actions that are not described as allowed (such actions could be illegal, but could also be not considered a priori), the participants can (e.g., through exploration) come up with more effective ways of doing things and react to unexpected situations which were not taking into consideration when the organizational regulations were recorded. In this case, however, the safety of the system has to be guaranteed by sanctioning participants for doing illegal actions.

Throughout this paper we will use a generic, simple example based on a simple regulation to *regulate the temperature of the building at a comfortable level without wasting energy.* This can be seen or described as the regulation or norm

that the organization has to comply to; namely, the thermostat is *obliged* to keep a comfortable level of heat in the building without wasting energy. Before we translate this regulation into service specifications, combining the services needed to achieve this objective, we first create an operational description of the general objective. That is to say, we give an operational meaning to the regulation, which is informing the organization, still on an abstract level but more concrete than the norm/regulation itself, how the objective is to be reached. There are alternative mappings to operational descriptions possible for this example regulation. For now, let us assume that the operational meaning of the regulation is that *the temperature in the building needs to be 18 °C whenever there are people around*. Finally, this operational description of the regulation is used to combine the services (at the implementation level) in such a way that the regulation of the organization can be fulfilled. In this case, this might mean the combination of the following services:

- a service to get the day of the week;
- a service to get the time of day;
- services to get the temperature of every room in the building;
- a service to translate temperatures in °Fahrenheit to °Celsius;
- a service to regulate the heater/air-conditioner of the building.

The services for the time of day and day of the week are needed to determine whether there are people in the building (i.e., the system does not need to use the heater/air-conditioner during evenings and weekends). The translation service from °F to °C is only needed if not all services (that measure the temperature or regulate the heater/air-conditioner) are speaking the same "language".

In this paper we compare the approach of ALIVE, based on previous research done [1, 3, 13, 14], to the regulation of organizations in general. In the next section we give a broad overview of the ALIVE project. In section 3, we explain how the use of intermediate levels helps the deployment of regulations. We present our ideas about how the transition of regulations from an abstract organizational point of view can be made to the concrete practice. Moreover, we present our ideas about how to cope with adaptivity and how this affects organizational structures. We end the paper with some conclusions.

2 The ALIVE approach

New generations of networked applications based on the notion of software services that can be dynamically deployed, adjusted and composed will make it possible to create radically new types of software systems. In turn, this will require profound changes in the way in which software systems are designed, deployed and managed – exchanging existing, primarily top-down "design in isolation" engineering, to new approaches which are based on integrating new functionalities and behaviours into existing running systems already active, distributed and interdependent processes.

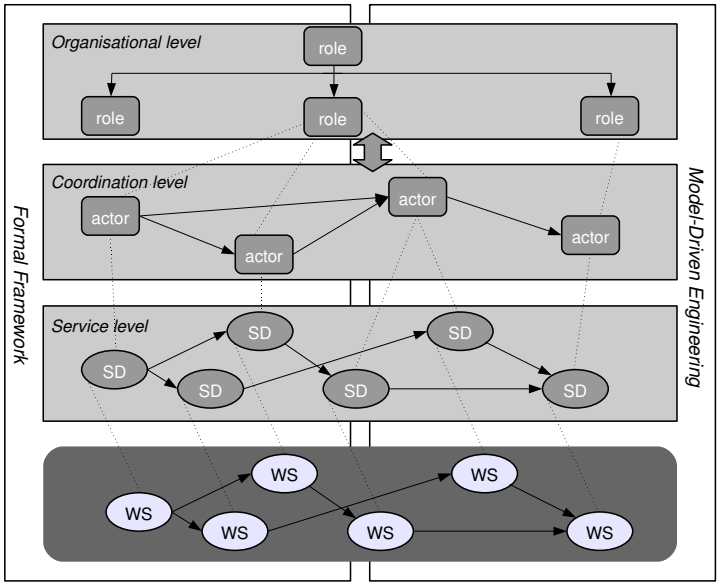

Fig. 1. The ALIVE framework for software and service engineering.

The ALIVE project is based around the central idea that many strategies used today to organize the vastly complex interdependencies found in human social, economic behaviour will be essential to structuring future service-based software systems. More specifically, the project aims to combine cutting edge Coordination and Organization mechanisms and Model Driven Design to create a framework for software and service engineering for "live" open systems of active services.

The project extends current trends in service-oriented engineering by adding three extra layers (see Figure 1).

- The *Service Layer* augments and extends existing service models with semantic descriptions (SD) to make components aware of their social context and of the rules of engagement with other web services (WS).
- The *Coordination layer* provides the means to specify, at a high level, the patterns of interaction between services, using a variety of powerful coordination techniques from recent European research in the area.
- The *Organization Layer* provides context for the other levels – specifying the organizational rules that govern interaction and using recent developments in organizational dynamics to allow the structural adaptation of distributed systems over time.

In the following sections we focus mainly on the connections between the organizational level (where the regulations reside) and the service level by using

an intermediate level. We show how the ideas of ALIVE relate to the deployment of regulations in human organizations and allow for flexible adaptation.

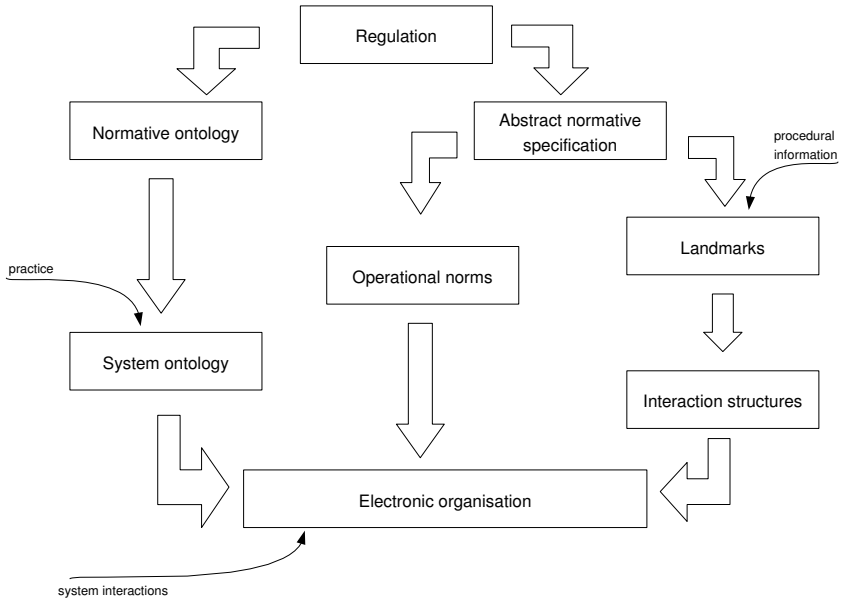

Fig. 2. From laws to electronic organizations.

3 From Abstract Regulation to Implementation

The deployment of regulations in the ALIVE approach consists of a gradual transition from the organizational level to the service-based implementation (the bottom two levels of the model in Figure 1). Given that organizations are characterized by their rules and conventions [1, 3], this process of implementing organizational regulations is then as proposed in Figure 2.

First, a formal representation of the regulations is created, giving an *abstract normative specification* of the allowed interactions in the organization (e.g., in deontic logic). Given our example, this means a formalization like, e.g., $O_{thermo}(temperature(comfortable))$ and $F_{thermo}(waste(energy))$. Which states that *thermo* is obliged to make sure that the temperature is comfortable and *thermo* is forbidden to waste energy. The creation of a formal representation of the regulations also creates a basis for the ontology that is needed (we call this basis the *normative ontology*). The normative ontology is built from: 1) the concepts and relations used in the formalization step, and 2) information taken

from the ontological definitions in the regulations themselves. In our example, the normative ontology contains the concepts of $comfortable$, $temperature$, $energy$, etc. The normative ontology and the formal representation of the regulations can be seen as the organizational level of Figure 1.

The normative specification is also used as the basis of the implementation of the regulations. The process from normative specification to implemented norms is as follows: 1) the abstract norms are translated to concrete *operational norms* (although these are only useable for a certain context, i.e., this particular organization, and less expressive than abstract norms, concrete norms are a lot easier to implement); 2) the operational norms are translated into constraints and procedures that will see to it that the norm is enforced in the organization. In our example, the operational norm is *the temperature in the building needs to be 18 °C whenever there is people around.* The design of *interaction structures* that can be used in the organization consists of the following steps: 1) the important characteristics of the norms that express how interactions should be in the organization are extracted from the norms to create a prototypical interaction structure on a high level of abstraction (we call these important steps derived from the norms *landmarks*, and the structure that expresses the ordering over these landmarks a *landmark pattern*); 2) by using procedural information and the expected capabilities of the system components an *interaction structure* is created to give a default manner for achieving certain objectives in the organization. For our example we can create an interaction structure by using landmarks (e.g., $L_1 =$*check temperature* and $L_2 =$*adjust heater*, with the temporal ordering that $L_1 < L_2$): $if(today = normal\_weekday)\ then\ temperature := requestTemperature(service_{temp});\ if(temperature \leq 18)\ then\ turnHeaterOn.$ The operational norms and landmark patterns provide the intermediate level of the transition from organizational regulations to a implementation. This level can be seen as the coordination level as shown in Figure 1.

Finally, the *system ontology*, which contains all concepts used in the norms as well as those used in the implementation is build from the normative ontology. The normative ontology is extended with the concepts and relations that follow from the operational and procedural information that was added to create the operational norms and the interaction structures. Moreover, concepts describing the system states and actions need to be added and linked as well.

As shown in Figure 2, the following four elements are of prime importance when implementing regulations in organizations:

- A common **ontology**, defining the meaning of concepts, the roles used in the organization and the relations between different contexts.
- A **normative specification** of the allowed interactions in the organization.
- **Interaction structures** to specify conventions in procedure mechanisms, giving a typical interaction profile which should work in any circumstance.
- An active **enforcement mechanism** to make sure that the participants of the organization adhere to the normative specification.

The ontology is needed to specify how the participants interact, defining the communicative propositions that are used, and defining the roles and role hi-

erarchy that is used throughout the norms. The normative specification is the basis of the organization, specifying the legal and illegal actions in the environment. Denoted in a formal language, this specification can be used to derive the last two elements of the framework. The interaction structures define standard ways in which the legal interactions can take place in the organization. They provide a means for non-norm aware participants to perform their task in the organization, or provide a guideline for norm-aware participants to follow (to show how things can be done, though are not necessarily the only way to do it, and can be deviated from if need arises). The norm enforcement is necessary to guarantee the safety of the system. Since we do not restrict the participants of the organization to only perform the allowed actions, the organization is required to check and enforce the proper ways of acting upon the participants in the organization. Much like in the real-world, instead of equipping all cars with speed-limiting devices, one specifies that speeding is illegal, and checks whether everyone adheres to that norm (even if one would opt for the regimented option of installing speed-limiting devices in cars, one would still have to check that no one tampers with the device and violates the norm).

An important step of this deployment process is the addition of operational information (taken from practice or procedures) to create an intermediate level (in Figure 2; the *operational norms* and the *landmarks*) that tries to capture the essence of the organizational level, but brings it closer to the actual implementation. Let us look at the addition of such information in more detail.

Adding Operational Information

The translation from organizational regulations from natural language to a formal representation (the abstract normative specification) is only the first step of the process of implementing the regulations. Usually the regulations are expressed at a high level of abstraction, to allow the regulation to cover a wide variety of situations and to be used for an extensive period of time without the need for modifications, it is hard to link these regulations to the concrete situations that arise in the practice. To make the normative specification useful in the deployment of the organization, an interpretation of the norm is needed, which should contain concrete (organizational) meanings of the vague and abstract terms used in the norm and which possibly contains procedural information that can be used to simplify the enforcement of the norm. This process of interpreting the norms to make them useable for a single context, i.e., the organization, is referred to as *contextualization* [1].

The contextualization process is meant to give a link between the abstract terms and concepts used in the abstract normative specification and the concrete situations and concepts that exist in the practice. Where norms contain terms such as 'fair' and talk about actions like 'discriminating', these concepts have no clear meaning in the implementation. There are, however, states and (sequences of) action(s) in the implementation that can be classified as an interpretation of one of these vague concepts in the *context of the organization*. These interpretations are highly context dependent and can differ from organization

to organization. For example, in accordance with the example described in the introduction, the abstract norm *regulate the heat of the building at a comfortable level without wasting energy* is contextualized (e.g., based on the preferences of the people that work in the buildings) to *the temperature in the building needs to be 18 °C whenever there are people around*. In another implementation, however, it could be something different, e.g., *the heater should be turned off at night or when the temperature is above 68 °F*.

Although norms that result from the contextualization process are concrete and contain only concepts that are meaningful in the organization, these norms still require further explicification before they can be implemented. Norms only have a declarative meaning, i.e., how things should be, while abstracting from operational meanings, which expresses how it should be achieved. Moreover, there is more than one way to enforce a single norm and procedural information (which is not part of the norm) will have to be used to decide how the norm is best implemented. This second translation process of adding additional operational and procedural information to the norms is referred to as *operationalisation* [1].

In the next section, taking advantage of recent results from adaptive system engineering approaches, we show our vision about how to cope with adaptivity requirements at the organizational level.

4 Introducing Adaptivity in organizations

Implementing regulations for organizations that are completely static is quite straightforward. The real challenge comes when the circumstances change and the organization needs to adapt to the new situation while still trying to abide by the regulations. In this section, we focus on those features of the proposed organization framework to effectively deal with context changes, namely, how the organizational models for the intended (service-based) system adapts to different kinds of changes. Before going into detail how our approach achieves adaptivity qualities, let us first look at how adaptivity is handled in other (recent) approaches.

A very compelling research topic within the area of software engineering regards methods, architectures, algorithms, techniques, and tools that can be used to support the development of adaptive systems. That is, software engineers are looking for techniques to model important requirements for adaptive software systems such as the ability to cope with changes of stakeholders' needs, changes in the operational environment, and resource variability. On one hand, a quite recent and interesting example of adaptive systems is IBM's work on autonomic software systems [5, 7]. Such a software type is characterised by properties of being able to automatically re-configure itself when new components come into or are removed from the system (self-configuration); being able to continually tune its parameters for optimisation (self-optimisation); being able to monitor, analyse, and recover from faults and failures when they occur (self-healing); and being able to protect itself from malicious attacks (self-protection).

On the other hand, promising software engineering approaches have recently adopted goal-oriented methodologies with an extensive use of goal models (GMs), which have been initially proposed in Distributed Artificial Intelligence as a means for capturing agent intentions and guiding agent coordination [6, 8] within dynamic environments. Within such methodologies, requirements are elicited, specified, and elaborated using the concept of goal, which can be used to model stakeholder and organizational objectives, but also an agent goal. In other words, the goal concept allows designer to represent high-level (strategic) concerns.

In [9, 11], GMs allow a designer to represent and reason about stakeholder objectives and agent goals in a given application domain in order to derive requirements for adaptive software. According to these approaches, GMs give support in exploring and evaluating alternative solutions which can meet stakeholders expectations (objectives) and in detecting conflicts that may arise from multiple viewpoints (see also [12]).

The above approaches identify several crucial components that a development framework should take into account to effectively deal with software adaptivity. Nevertheless, how such requirements affect organizational structures has not been completely addressed. Finally, in [2, 4] interesting approaches to cope with reorganization issues have been presented. The principal aim in [2, 4] has been to develop a modelling language to describe organizational structures, and how their objectives are related to changes in the environment. Specifically, a simulator framework, where agents play modeled organizational roles having different objectives, has been adopted to test reorganization behaviours to cope with changes.

Adaptivity within organizations

Results from the above approaches, related to specifying the system adaptivity, are useful to properly interpret and reflect such requirements at the organizational level. Specifically, we aim at illustrating by simple example scenarios that our framework (see Figure 1) can distribute the complexity –to handle context changes– among its different layers. This latter property is important to improve the flexibility and robustness of the (service-based) system. Adaptations on the lower level might violate specific procedural interpretations of a regulation, but still comply to the more abstract regulation specified on a higher level. When such a situation occurs one can now change the operationalizaion of the regulation such that the new practices conform to these procedures, while preserving the same regulation at an abstract level.

Principal sources/causes of dynamic changes in the context can be described as follows.

Stakeholder needs. Changes in the stakeholder needs happen frequently in open organizations where new roles may be added and old ones are detached in order to better reflect the market changes. In other words, stakeholder needs have to be strictly related with organizational objectives to effectively deal with changes of needs, e.g., re-adapting to new organization market strategies. Ac-

cording to our example, let us assume an enterprise [1] has to pursue the objective *make employees comfortable* and to do that it depends on the work and quality of thermostat devices of all departments provided by a thermostat organization. Then, let us consider that because of the market strategy, the area-manager changes her needs, delegating to each department-manager the objective *minimise heating costs* to pursue too, which requires reorganising its internal service providing structure. This change may result in selecting another service to play the role *thermostat* that is cheaper, by e.g. auctioning a new offer. Notice that, according to Figure 1, this change is sensed at organizational level but handled at coordination level.

Environment conditions. Depending on the kind of application domain, such requirements have to reflect real life situations into the organizational behaviour, e.g., symptoms to be forecasted (at design-time) and then anticipated (at runtime) to avoid failures in pursuing objectives. Moreover, such requirements deal also with how norms can affect and are related to organization objectives. According to our example, let us assume that the thermostat has a digital power meter (*services to get the temperature of every room in the building*) in order to maintain its objective *regulate the heat of the building to a comfortable level without wasting energy*. This service periodically verifies whether the consumed energy in each building correctly stays into a specific range. Let us also assume that, during the winter time, a couple of employees went on holiday but forgot to close their office windows. This environment change (symptom) could cause the failure of the previous objective (expressed in the normative specification) if no countermeasures (enforcement mechanisms) have been considered in advance to properly handle such a fault symptom. The enforcement has to trigger another objective achievement, e.g., asking to the building attendant to check all the windows, therefore such a change mainly affects the coordination level of Figure 1. Moreover, to get such a process completely automated, the reasoning mechanisms have to be supported by (domain) ontologies that describe symptoms, faults, recovery objectives, and roles along their relationships.

System functionalities. Although the modelling of the organizational knowledge level has a key role within the whole framework, role and objective concepts need to be properly grounded into specific system functionalities (agent capabilities and/or service functionalities) in order to really affect and sense the environment. In other words, changes in environment and in stakeholder needs (discussed above) inherently are reflected in the orchestration process of the service level of Figure 1, following an implicit top-down approach (see Figure 2). In the other hand, changes in system functionalities deal with a bottom-up propagation, namely, several dynamic issues can arise from the service-level and, consequently, need to be related and handled by the organizational and coordination levels. Let us consider the example sketched in Section 1, where the thermostat has to maintain the regulation *the temperature in the building needs*

[1] According to the example of Section 1, this enterprise acts as the committer for the thermostat organization (supplier), i.e., roles commonly played within any organization.

to be 18 °C whenever there are people around (O_1). To achieve this objective, the information system has to orchestrate different services and then combine their results collected every time, e.g., get the day of the week ($s_1^{O_1}$), get the time of day ($s_2^{O_1}$), translate temperatures from °Fahrenheit to °Celsius ($s_3^{O_1}$) because the thermostat device works in °Fahrenheit, calculate whether the sensed temperature is in the established range ($s_4^{O_1}$), and sense the environment for people presence ($s_5^{O_1}$). Now, let us assume that at the time O_1 had to be achieved, the system recognizes that $s_3^{O_1}$ is not available. *Where and how to handle this sensed change?* Maybe the service level (the *where*) has been provided with some simple recovery function (the *how*) such as searching for an equivalent service. But, the most compelling scenario arises when the service level brings about some important failure (e.g. no other equivalent service available), propagating it to the next-up level. Again using different levels of abstraction in the specification now allows for different solutions using different types of knowledge present at those levels.

5 Conclusions and Future Work

In this paper we presented how the ALIVE approach can be used for the deployment of regulations and the reorganization of both human societies and information systems. The ALIVE project aims to create a framework that combines coordination and organization mechanisms in order to provide flexible, high-level models to assist software and service engineering. A main element of the ALIVE approach is to distribute the design of coordination and organization of service-based implementations over different levels of abstraction. At the highest level of abstraction (the organizational level) the context is defined in terms of abstract regulations and objectives. These abstract norms are operationalized in the next level of abstraction (the coordination level), where operational and contextual information is added taken from procedures and practice. The lowest level of abstraction (the service level) then deploys the operational norms and structures defined on the coordination level to create a service-based implementation.

The process of translating the abstract regulation to an implementation has been illustrated. In this process, the main elements are 1) an abstract normative specification, 2) a common system ontology, 3) a set of interaction structures describing default interactions, and 4) mechanisms for enforcing the norms to guarantee safety in the system. Moreover, we have shown that the translation or regulations to an implementation in practice is not a straight-forward, one-step process. The organizational regulations have to be *contextualized* and *operationalized* before they can be implemented. That is, the abstract regulations need to be translated into more concrete regulations which use only concrete concepts and relations (which are context dependent) and operational information, to express how the regulation is to be achieved/maintained, has to be added.

This paper also reports on how the proposed framework naturally fits for the modelling of context (requirements) changes to better reflect real organization behaviours. Moreover, taking advantage from system specification of self-

adaptive systems, we have shown how the framework can handle such requirements at organizational and coordination levels.

As future work, we are interesting to investigate how the organizational framework should behave to deal with context changes that are not within the organization knowledge, e.g., new objectives, roles, and relationships not described in the domain ontology. This challenging research aspect is very related to both the evolutionary design and the evolutionary qualities of agent systems, namely, how to automatically update organization models from new knowledge that emerges from the service level.

References

1. H. Aldewereld. *Autonomy vs. Conformity: an Institutional Perspective on Norms and Protocols*. PhD thesis, Universiteit Utrecht, June 2007.
2. F. Dignum, V. Dignum, and L. SonenBerg. Exploring congruence between organizational structure and task performance: a simulation approach. In *Coordination, Organisation, Instiutions and Norms in Agent Systems I*, LNAI 3913, 2006.
3. V. Dignum. *A Model for Organizational Interaction: based on Agents, founded in Logic*. PhD thesis, Universiteit Utrecht, 2004.
4. V. Dignum and C. Tick. Agent-based Analysis of Organizations: Performance and Adaptation. In *2004 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2007)*, California, USA, 2007. IEEE CS Press.
5. A. G. Ganek and T. A. Corbi. The dawning of the autonomic computing era. *IBM Systems Journal*, 42(1):5–18, 2003.
6. N. Jennings. *Foundations of Distributed Artificial Intelligence*, chapter Coordination Techniques for Distributed Artificial Intelligence. Wiley-IEEE, 1996.
7. J. O. Kephart and D. M. Chess. The vision of autonomic computing. *Computer, IEEE Computer Society Press*, 36(1):41–50, 2003.
8. V. Lesser. A retrospective view of fa/c distributed problem solving. In *Systems, Man and Cybernetics, IEEE Transactions on*, volume 21, pages 1347–1362. 1991.
9. M. Morandini, L. Penserini, and A. Perini. Towards Goal-Oriented Development of Self-Adaptive Systems. In *Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2008)*. ACM and IEEE digital libraries, to appear, 2008.
10. A. Omicini, A. Ricci, M. Viroli, C. Castelfranchi, and L. Tummolini. Coordination artifacts: Environment-based coordination for intelligent agents. In *Proc. of the 3rd Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS 2004)*, pages 286–293. ACM Press, 2004.
11. L. Penserini, A. Perini, A. Susi, and J. Mylopoulos. High Variability Design for Software Agents: Extending Tropos. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 2(4), 2007.
12. A. van Lamsweerde and E. Letier. Handling obstacles in goal-oriented requirements engineering. *IEEE Transactions on Software Engineering, Special Issue on Exception Handling*, 26(10), 2000.
13. J. Vázquez-Salceda. *The Role of Norms and Electronic Institutions in Multi-Agent Systems. The HARMONIA framework*. Whitestein Series in Software Agent Technology. Birkhäuser Verlag, 2004.
14. J. Vázquez-Salceda, V. Dignum, and F. Dignum. Organising multiagent systems. *JAAMAS*, 11(3):307–360, November 2005.